

1ο ΕΠΑΛ Περάματος -7ο ΕΚ Πειραιά

# Εφαρμογές Arduino

Σεμινάριο Ηλεκτρονικού Τομέα



Φεβρουάριος 2014

## Εισαγωγή

Με αυτό το σεμινάριο φιλοδοξούμε να μάθουμε ο ένας στον άλλο βασικές αρχές και εφαρμογές που αφορούν την πλατφόρμα arduino.

Σχεδιάστηκε από την Σύμβουλο των Ηλεκτρονικών (ΠΕ12.10) Κα Σαλωνίδου Αθηνά και τον συνάδελφο Μαλτέζο Γιάννη, ενώ στην επιμέλεια του υλικού συμμετείχαν οι συνάδελφοι Νίκας Αναστάσιος, Κουβαράκης Γιάννης και Δερμάτης Χαράλαμπος.

Γενική φιλοσοφία του σεμιναρίου είναι η πρακτική εξάσκηση (Hands on Training). Κάθε αντικείμενο θα συζητηθεί μέσω ασκήσεων και παραδειγμάτων χρήσιμων για την κατανόηση και σωστή χρήση του arduino.

Το σεμινάριο έχει σχεδιαστεί έτσι ώστε κάποιες βασικές γνώσεις να μην είναι απαραίτητες, μιας και η προσέγγιση του αντικειμένου γίνεται από την αρχή.

Οι σημειώσεις είναι προϊόν επεξεργασίας των πηγών που αναφέρουμε και τα πνευματικά δικαιώματα ανήκουν στους δημιουργούς τους.

Σε όλες τις πρακτικές εφαρμογές έχει γίνει πειραματική επαλήθευση στο εργαστήριο Ηλεκτρονικών του σχολείου μας.

Το θεωρητικό μέρος του υλικού είναι από την πτυχιακή εργασία των σπουδαστών του ΤΕΙ Σερρών Φελλόπουλου Αναστάσιου και Σπύρου Μαρίας με επιβλέποντα τον Δρ. Σταύρο Βολογιαννίδη, που αφορά την κατασκευή ενός τρίτροχου κινούμενου ρομπότ με πλατφόρμα Arduino .

Από την πτυχιακή χρησιμοποιήθηκαν τα κεφάλαια 1,2 και 8 αυτούσια και χωρίς καμία επέμβαση στο υλικό των δημιουργών τους που ευχαριστούμε θερμά



Τεχνολογικό Εκπαιδευτικό Ίδρυμα Σερρών  
Σχολή Τεχνολογικών Εφαρμογών  
Τμήμα Πληροφορικής και Επικοινωνιών

Υλοποίηση ενός τρίτροχου κινούμενου ρομπότ χρησιμοποιώντας  
πλατφόρμα Arduino

### **Πτυχιακή Εργασία**

Φελλόπουλος Αναστάσιος ΑΜ: 1989  
Σπύρου Μαρία ΑΜ: 2084

Επιβλέπων: Δρ. Σταύρος Βολογιαννίδης, Επιστημονικός Συνεργάτης

ΣΕΡΡΕΣ, ΜΑΪΟΣ 2012

# Περιεχόμενα

Περιεχόμενα.....	vi
ΚΕΦΑΛΑΙΟ 1 .....	1
Arduino .....	1
ΚΕΦΑΛΑΙΟ 2.....	4
Arduino Uno.....	4
2.1 Γενικά.....	4
2.2 Μέρη ενός Arduino Uno .....	4
2.3 Σειριακή Θύρα .....	5
2.4 Χαρακτηριστικά του Arduino .....	6
2.5 Βασικές μνήμες .....	6
2.6 Τροφοδοσία .....	7
2.7 Επικοινωνία .....	8
2.8 Γλώσσα Προγραμματισμού.....	8
2.9 Εγκατάσταση του προγράμματος.....	8
2.10 Ολοκληρωμένο Περιβάλλον Ανάπτυξης του Arduino.....	13
2.11 Σειριακή οθόνη (Serial Monitor).....	14
2.12 Η Δομή το προγράμματος .....	15
2.13 Βασικές δομές και λειτουργίες προγραμματισμού .....	15
2.14 Ψηφιακές ακίδες (Digital pins) .....	18
2.15 Αναλογικές ακίδες εισόδου (Analog input pins).....	19
2.16 Υποστήριξη Βιβλιοθηκών (Libraries) .....	20

ΚΕΦΑΛΑΙΟ 8.....	100
Processing .....	100
8.1 Χαρακτηριστικά της Processing.....	100
8.2 Έργα βασισμένα στην Processing .....	100
8.3 Η δομή του προγράμματος .....	101
8.4 Βιβλιοθήκες και εργαλεία .....	102
8.5 Πρόγραμμα 5: Αναπαράσταση αποτελεσμάτων σε πραγματικό χρόνο.....	103
ΚΕΦΑΛΑΙΟ 9.....	110
Βιβλιογραφία – Πηγές πληροφοριών.....	110

# ΚΕΦΑΛΑΙΟ 1

# Arduino



Οι πλατφόρμες Arduino κατασκευάζονται κυρίως από την εταιρία Smart Project. Ωστόσο, το Arduino ξεκίνησε ως έργο προς ανάπτυξη το 2005 στην Ιταλία, στο Ινστιτούτο Αλληλεπίδρασης Σχεδίασης Invea ώστε οι φοιτητές του Ινστιτούτου να αναπτύσσουν ενσωματωμένα συστήματα οικονομικά και αποδοτικά αξιοποιώντας τις δυνατότητες και τις ευκαιρίες που μπορεί να προσφέρει το ελεύθερο λογισμικό.

Γενικότερα, το Arduino θα λέγαμε ότι είναι ένα εργαλείο που μπορούμε να κατασκευάσουμε ένα υπολογιστικό σύστημα με την έννοια ότι αυτό θα ελέγχει συσκευές του φυσικού κόσμου, σε αντίθεση με τον κοινό Ηλεκτρονικό Υπολογιστή. Βασίζεται σε ευέλικτο, εύκολο στη χρήση υλικό και λογισμικό, σε μια αναπτυξιακή πλακέτα που ενσωματώνει επάνω έναν μικροελεγκτή και συνδέεται με τον Η/Υ για να προγραμματιστεί μέσα από ένα απλό περιβάλλον ανάπτυξης. Με το Arduino δημιουργούνται συσκευές οι οποίες εξυπηρετούν διάφορους σκοπούς έχοντας την δυνατότητα να δέχονται ερεθίσματα από το περιβάλλον τους (μέσω των αισθητήρων) και να αντιδρούν ανάλογα με το πως έχουν προγραμματιστεί.

Τα παραπάνω δεν ακούγονται πρωτότυπα. Υπάρχουν και άλλες πλατφόρμες και υλοποιήσεις που μπορούν να κάνουν τα ίδια πράγματα. Ποια είναι η ειδοποιός διαφορά; Το Arduino βασίζεται σε τεχνολογίες ανοιχτού κώδικα. Μπορεί να κατασκευαστεί από τον καθένα, μπορεί να ενσωματωθεί σε συσκευές ακόμα και για εμπορικούς σκοπούς και το σημαντικότερο είναι ότι υπάρχει μια ολόκληρη κοινότητα που χρησιμοποιεί το Arduino σε κατασκευές άρα υπάρχει μεγάλος όγκος ελεύθερης πληροφορίας. Γενικά, τα Projects στον εν λόγω Μικροελεγκτή μπορούν να είναι αυτόνομα (σε επίπεδο hardware) ή να επικοινωνούν με κάποιο software στον Η/Υ του προγραμματιστή (προγράμματα όπως τα Flash, Processing, MaxMSP). Το Arduino χρησιμοποιεί τώρα ένα ειδικά προγραμματιζόμενο Atmega382 αντί του chip FTDI ώστε αυτό να επιτρέπει τόσο την πιο γρήγορη ταχύτητα μεταφοράς όσο και τη γρήγορη σειριακή επικοινωνία.

Ο μικροεπεξεργαστής ενός Arduino συνήθως προγραμματίζεται εκ των προτέρων ώστε να παρέχει κάποιο φορτωτή εκκίνησης (BootLoader). Ο φορτωτής εκκίνησης υπάρχει ώστε να απλοποιεί την διαδικασία της αποθήκευσης των προγραμμάτων στην Flash Memory του Arduino μέσω σειριακής USB θύρας.

Επιπλέον, η γλώσσα προγραμματισμού, οι διάφορες βιβλιοθήκες και το ολοκληρωμένο περιβάλλον ανάπτυξης που υπάρχουν για τον προγραμματισμό της πλατφόρμας Arduino αποτελούν ανοιχτό λογισμικό προσφέροντας έτσι ανεκτίμητη γνώση σε όλους.

### Βασικά Πλεονεκτήματα πλατφόρμας Arduino:

- **Οικονομική:** Η πλατφόρμα Arduino αποτελεί οικονομική λύση διότι είναι φθηνότερη. Επιπλέον, είναι αρχιτεκτονικά ανοιχτή και μπορεί ο οποιοσδήποτε να την αναπτύξει από μόνος του.
- **Μεταφέρσιμη:** Σε σχέση με τις υπάρχουσες πλατφόρμες στο εμπόριο η πλατφόρμα Arduino παρέχει πλήρη μεταφερσιμότητα με αποτέλεσμα να μπορεί να προγραμματιστεί στα περισσότερα λειτουργικά συστήματα.
- **Επεκτάσιμη:** Το υλικό και το λογισμικό της πλατφόρμας Arduino είναι ανοιχτά και ελεύθερα για όλους. Καθημερινά, χιλιάδες υποστηρικτές του ελεύθερου λογισμικού αναπτύσσουν διάφορες βιβλιοθήκες για την υποστήριξη της πλατφόρμας. Παράλληλα, τόσο η αρχιτεκτονική όσο και το υλικό της πλατφόρμας εξελίσσονται συνεχώς.

Παρακάτω ακολουθούν μερικές από τις πλατφόρμες Arduino που έχουν αναπτυχθεί και όπου η κάθε μία είτε αποτελεί εξέλιξη κάποιας άλλης, είτε έχει αναπτυχθεί για κάποιο συγκεκριμένο σκοπό :

- Arduino Uno
- Arduino Diecimila
- Arduino Duemilanove
- Arduino Mega1280
- Arduino Mega2560
- Arduino Mini
- Arduino Nano
- Arduino USB
- Arduino Stamp
- Arduino Fio
- Arduino NG
- Arduino NG+
- Arduino Extreme
- Arduino Bluetooth
- LilyPad Arduino
- Serial Arduino

Ακολουθεί ένας πίνακας όπου περιέχει για τις πιο τυπικές πλατφόρμες Arduino τα βασικά χαρακτηριστικά όσο αφορά το υλικό τους μέρος.

Πλατφόρμα Arduino	Μικροελεγκτής Atmel AVR	Flash KiB	EEPROM KiB	SRAM KiB	Ψηφιακές Επαφές E / E	PWM	Αναλογικές Επαφές Εισόδου
<b>Diecimila</b>	ATmega168	16	0.5	1	14	6	6
<b>Duemilanove</b>	ATmega168/328	16	0.5	1	14	6	6
<b>Uno</b>	<b>ATmega328</b>	<b>32</b>	<b>1</b>	<b>2</b>	<b>14</b>	<b>6</b>	<b>6</b>
<b>Mega</b>	ATmega1280	128	4	8	54	14	16
<b>Fio</b>	ATmega328P	32	1	2	14	6	8
<b>Mega 2560</b>	ATmega2560	256	4	8	54	14	16

## 1.1 Το λογισμικό του Arduino

Το περιβάλλον ανάπτυξης του Arduino (IDE) έχει συγγραφεί με την γλώσσα προγραμματισμού Java και αυτό το καθιστά μεταφέσιμο στα περισσότερα λειτουργικά συστήματα. Το IDE του Arduino περιέχει έναν έξυπνο συντάκτη, μεταγλωττιστή της C, C++, τερματικό για σειριακή επικοινωνία με το Arduino, κ.α.

Πιο συγκεκριμένα, η γλώσσα προγραμματισμού που χρησιμοποιείται για την συγγραφή προγραμμάτων στο Arduino είναι η Wiring (C, C++). Το IDE του Arduino χρησιμοποιεί εργαλεία GNU toolchain και AVR Libc για να παρέχει την μεταγλώττιση προγραμμάτων από C, C++ σε κατάλληλες AVR εντολές γλώσσας μηχανής, καθώς και το εργαλείο avrdude για την αποστολή του εκτελέσιμου προγράμματος στην Flash memory του Arduino.

Η ψηφιακή σχεδίαση του υλικού μέρους του Arduino είναι ανοιχτή και προσβάσιμη από όλους μια και είναι δημοσιευμένη υπό την άδεια Creative Commons Attribution Share-Alike 2.5. Επίσης, το περιβάλλον ανάπτυξης (IDE) του Arduino είναι ελεύθερο λογισμικό και είναι δημοσιευμένο υπό την άδεια GNU General Public License Version2.



# ΚΕΦΑΛΑΙΟ 2

## Arduino Uno

### 2.1 Γενικά

Η καρδιά του Arduino Uno είναι φυσικά ένας μικροεπεξεργαστής. Αυτός είναι το «μυαλό» του Arduino και είναι προγραμματιζόμενος ώστε να ελέγχει τα 14 ψηφιακά input/output pins και τα 6 αναλογικά που υπάρχουν πάνω στην πλακέτα ανάπτυξης. Δια μέσου αυτών των 20 pins γίνονται όλες οι διασυνδέσεις με εξωτερικά στοιχεία (κινητήρες, LEDs, LCD οθόνες κλπ) και αισθητήρες (Ultrasonic, θερμομέτρα, accelerometers κ.α).

Στην πλακέτα ανάπτυξης υπάρχει μια θύρα USB. Μέσω αυτής γίνεται η μεταφορά δεδομένων από αυτήν προς κάποια άλλη συσκευή, συνήθως έναν υπολογιστή, και το αντίστροφο. Ωστόσο, η κύρια χρήση στα αρχικά στάδια εκμάθησης είναι η μεταφορά του προγράμματος από τον υπολογιστή στον μικροεπεξεργαστή αλλά και η οπτικοποίηση των δεδομένων που απορρέουν από την λειτουργία της συσκευής μετά από το προγραμματισμό.

### 2.2 Μέρη ενός Arduino Uno

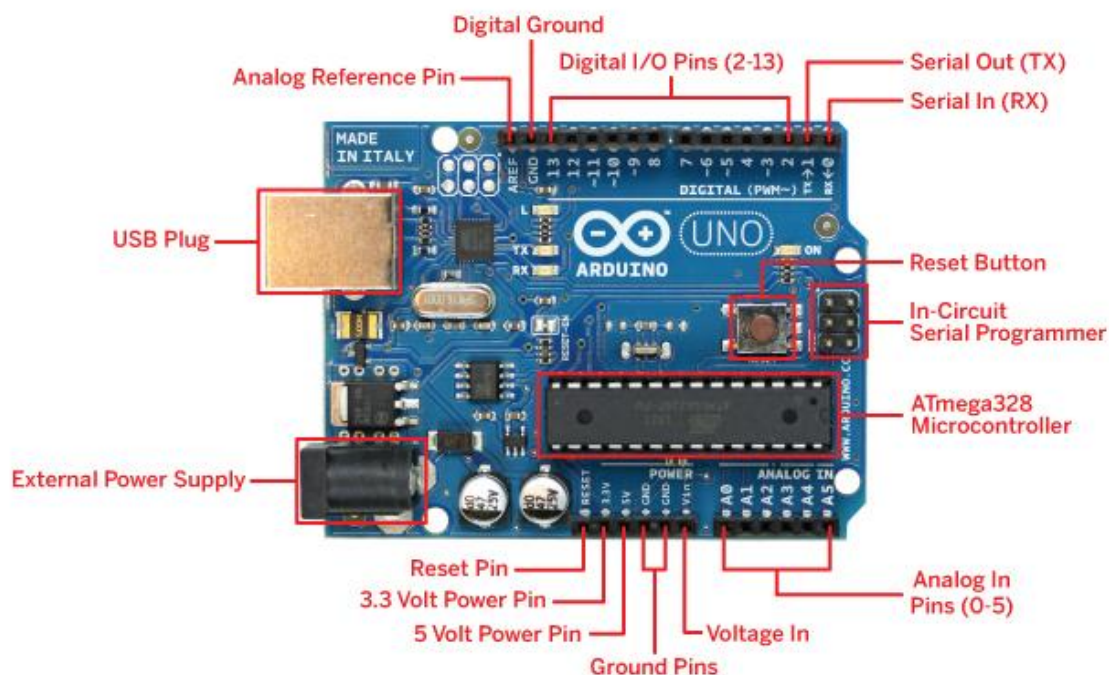
Το Arduino έχει 14 ψηφιακούς ακροδέκτες Εισόδου/Εξόδου οι οποίοι μπορούν να τεθούν ως είσοδοι ή ως έξοδοι με τις εντολές-συναρτήσεις pinMode(), digitalWrite(), και digitalRead() που θα αναλυθούν παρακάτω. Λειτουργούν στα 5 Volts και έχουν την δυνατότητα να παρέχουν ή να καταβυθίζουν ένταση της τάξεως των 40mA. Σε κάθε pin υπάρχει εσωτερικά ένας Pull-up αντιστάτης στα 20-50KΩ. Επιπλέον, έχει 6 αναλογικούς ακροδέκτες Εισόδου. Αυτοί μπορούν να διαβάσουν αναλογικές τιμές όπως η τάση μιας μπαταρίας κτλ και να τις μετατρέψουν σε έναν αριθμό από 0-1023. Η μέτρηση της τάσης γίνεται από προκαθορισμένα από 0 έως 5 volts. Εκτός αυτού, 6 εκ των 14 ψηφιακών ακροδεκτών οι P3, P5, P6, P9, P10 και P11 έχουν την δυνατότητα να προγραμματιστούν ώστε να λειτουργούν ως αναλογικές Έξοδοι.

Κάποιοι ακροδέκτες έχουν συγκεκριμένες λειτουργίες.

- Σειριακή Λειτουργία: 0 (RX) και 1 (TX). Χρησιμοποιούνται για λήψη (RX) και εκπομπή (TX) TTL σειριακών δεδομένων.
- Εξωτερικές Διακοπές: 2 και 3. Αυτοί οι ακροδέκτες μπορούν να ενεργοποιούν διακοπές αν ανιχνευθεί παλμός χαμηλής τάσης. Με την συνάρτηση

attachInterrupt(). Ο σκανδαλισμός των διακοπών μπορεί να γίνεται στο λογικό 0,1.

- PWM: 3, 5, 6, 9, 10, και 11 pins. Παρέχουν Έξοδο 8-bit PWM με την συνάρτηση analogWrite().
- SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Αυτοί οι ακροδέκτες επιτρέπουν επικοινωνία SPI, η οποία αν και παρέχεται από το hardware δεν είναι ακόμα διαθέσιμη στην γλώσσα προγραμματισμού του Arduino.
- LED: 13. Στον ακροδέκτη 13 υπάρχει ένα ενσωματωμένο LED. Όταν ο ακροδέκτης έχει τιμή HIGH, το LED ανάβει ενώ όταν το pin είναι LOW δεν ανάβει.



Εικόνα 2.1 Μέρη της πλατφόρμας Arduino Uno

## 2.3 Σειριακή Θύρα

Χρησιμοποιείται για επικοινωνία μεταξύ της πλατφόρμας Arduino και ενός υπολογιστή ή με άλλες συσκευές. Επομένως, όλες οι πλακέτες έχουν τουλάχιστον μια σειριακή θύρα. Επικοινωνεί με τις ψηφιακές ακίδες 0 (RX) και 1 (TX), καθώς και με τον υπολογιστή μέσω USB. Έτσι, εάν χρησιμοποιείται αυτή η λειτουργία(USB), δεν μπορούν ταυτόχρονα να χρησιμοποιηθούν οι ακίδες 0 και 1 για ψηφιακή είσοδο ή έξοδο.

Αξίζει να αναφερθεί, η ενσωματωμένη σειριακή οθόνη στο περιβάλλον του Arduino μπορεί να χρησιμοποιηθεί για να επικοινωνεί με την πλακέτα Arduino. Κάνοντας κλικ στο κουμπί Serial Monitor στην γραμμή εργαλείων και επιλέγοντας την ίδια ταχύτητα που χρησιμοποιείται στην κλήση της Serial.begin().

Οι βασικές συναρτήσεις της σειριακής θύρας είναι:

- `begin()` (αρχικοποίηση της σειριακής)
- `end()` (κλείσιμο της σειριακής)
- `available()` (έλεγχος αν υπάρχουν δεδομένα να διαβαστούν)
- `read()` (ανάγνωση των εισερχόμενων σειριακών δεδομένων)
- `peek()` (επιστρέφει το επόμενο byte από την σειριακή)
- `flush()` (άδειασμα του buffer της σειριακής από δεδομένα που έχει)
- `print()` (γράφσιμο δεδομένων στη σειριακή)
- `println()` (το ίδιο με την `Print()`, αλλά με αλλαγή γραμμής στο τέλος)
- `write()` (γράφει δυαδικά δεδομένα στη σειριακή)

## 2.4 Χαρακτηριστικά του Arduino

- Microcontroller: ATmega328
- Τάση λειτουργίας: 5V
- Τάση εισόδου: 7-12V
- Τάση εισόδου (όριο): 6-20V
- Digital I/O Pins: 14 (εκ των οποίων 6 περιέχουν PWM εξόδους)
- Analog Input Pins: 6
- DC ρεύματος I/O Pin: 40 mA
- DC τρέχουσα για 3.3V Pin: 50 mA
- Flash Memory: 32 KB εκ των οποίων 0,5 KB που χρησιμοποιούνται από τον bootloader
- SRAM: 2 KB
- EEPROM: 1 KB
- Clock Speed: 16 MHz

## 2.5 Βασικές μνήμες

Οι πλατφόρμες Arduino διαθέτουν τρεις βασικές μνήμες:

- Flash memory (32 Kbytes) στην οποία τοποθετείται κάθε φορά το πρόγραμμα που πρόκειται να εκτελεστεί καθώς και ο φορτωτής εκκίνησης που διευκολύνει την διαδικασία του προγραμματισμού της πλατφόρμας.
- SRAM memory (στατική μνήμη τυχαίας προσπέλασης των 2 Kbytes) η οποία χρησιμοποιείται για την προσωρινή αποθήκευση των στατικών και των μεταβλητών δεδομένων του προγράμματος που εκτελείται.
- EEPROM memory (1 Kbytes) στην οποία αποθηκεύονται οι τιμές των μεταβλητών όταν η πλατφόρμα σβήσει(OFF). Χρησιμοποιείται για την αποθήκευση ρυθμίσεων και άλλων παραμέτρων ανάμεσα στα Reset του Arduino.

Πρέπει να προστεθεί, η μνήμη Flash και η μνήμη EEPROM είναι σταθερές (οι πληροφορίες παραμένουν μετά την απενεργοποίηση του ρεύματος). Η μνήμη SRAM είναι ασταθής και οι πληροφορίες χάνονται όταν εναλλάσσεται το ρεύμα.

Επειδή δεν υπάρχει πολύ διαθέσιμη SRAM, αν τελειώσει, το πρόγραμμα μπορεί να αποτύχει με απροσδόκητους τρόπους. Μπορεί να φαίνεται ότι φορτώνει με επιτυχία, αλλά δεν τρέχει, ή τρέχει παράξενα. Για να ελεγχθεί εάν αυτό συμβαίνει, μπορούν να μειωθούν τα σχόλια ή οι σειρές ή άλλες δομές δεδομένων στο sketch (χωρίς να αλλάξει ο κώδικας). Εάν λειτουργεί με επιτυχία στη συνέχεια, κατά πάσα πιθανότητα έχει εξαντληθεί η SRAM. Ένας τρόπος για να αντιμετωπιστεί αυτό το πρόβλημα είναι αν υπάρχουν πίνακες αναζήτησης ή άλλοι μεγάλοι πίνακες, τότε μπορεί να χρησιμοποιηθεί ο μικρότερος τύπος δεδομένων που είναι αναγκαίος για να αποθηκευτούν οι τιμές που χρειάζονται.

## 2.6 Τροφοδοσία

Το Arduino Uno τροφοδοτείται είτε από εξωτερική τροφοδοσία που παρέχεται είτε μέσω μιας υποδοχής των 2.1mm (θετικός πόλος στο κέντρο) που βρίσκεται στην κάτω αριστερή γωνία του Arduino είτε απευθείας από την θύρα USB του υπολογιστή. Η επιλογή της πηγής γίνεται αυτόματα από το αναπτυσσόμενο. Ως εξωτερική τροφοδοσία ορίζεται είτε μια μπαταρία, είτε μετασχηματιστής των 9Volt από 220V. Η μπαταρία μπορεί να συνδεθεί στις υποδοχές του Arduino Vin και GND όπου τοποθετούνται ο θετικός πόλος και ο αρνητικός αντίστοιχα. Από την άλλη αν τροφοδοτηθεί με μετασχηματιστή απλά πρέπει να τοποθετηθεί το βύσμα στην υποδοχή που υπάρχει θετικό πόλο στο κέντρο.

Η πλακέτα μπορεί να λειτουργήσει με εξωτερική πηγή από 6 έως 20 Volts. Αν ωστόσο τροφοδοτηθεί με λιγότερα από 7 Volt τα pin εξόδου 5Volt δεν θα καταφέρουν να εξάγουν τάση 5 Volts. Αντίθετα, αν δώσουμε πάνω από 12 Volts θα υπερθερμανθεί ο σταθεροποιητής τάσης στην πλακέτα και ενδεχομένως να καταστραφεί. Συνεπώς, μια ιδανική τάση είναι τα 9 Volts.

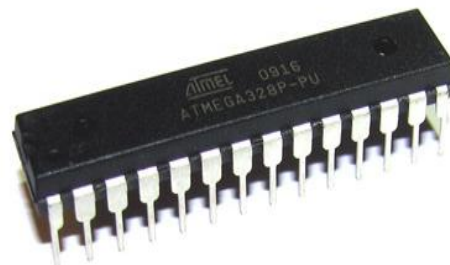
Οι ακροδέκτες τροφοδοσίας είναι οι εξής:

- **VIN.** Ακροδέκτης για μη σταθεροποιημένη τάση. Συνήθως εδώ συνδέεται μια εξωτερική πηγή τροφοδοσίας.
- **5V.** Ακροδέκτης σταθεροποιημένης τάσης 5Volt. Η ρυθμιζόμενη παροχή ηλεκτρικού ρεύματος που χρησιμοποιείται για την τροφοδοσία του μικροελεγκτή ή άλλων ηλεκτρονικών στοιχείων της πλακέτας. Αυτό μπορεί να προέρχεται είτε από Vin με ενσωματωμένο ρυθμιστή, ή να παρέχεται από USB ή άλλη ρυθμιζόμενη παροχή 5V
- **3V3.** Μέγιστη κατανάλωση ρεύματος είναι 50mA.
- **GND.** Γειωμένες ακίδες

## 2.7 Επικοινωνία

Το Arduino Uno έχει την δυνατότητα να επικοινωνεί με τον Ηλεκτρονικό Υπολογιστή, έναν άλλον Arduino ή άλλους μικροελεγκτές. Το ολοκληρωμένο ATmega328 παρέχει σειριακή επικοινωνία TTL 5 Volt UART, η οποία είναι διαθέσιμη από τους ακροδέκτες (λήψη RX) 0 και (εκπομπή TX) 1 του ολοκληρωμένου.

Επιπλέον, η αναπτυξιακή πλακέτα του Arduino παρέχει σειριακή επικοινωνία με τον Ηλεκτρονικό Υπολογιστή για προγραμματισμό με την βοήθεια ενός ειδικά προγραμματιζόμενου ενσωματωμένου ολοκληρωμένου ATmega328 αντί του chip FTDI. Ωστόσο, αυτό επιτρέπει την πιο γρήγορη ταχύτητα μεταφοράς και γρήγορης σειριακής επικοινωνίας. Με την σύνδεση του Arduino μέσω της θύρας USB αυτό εμφανίζεται ως εικονική σειριακή θύρα COM στο λογισμικό του υπολογιστή. Το firmware ATmega328 χρησιμοποιεί τα προγράμματα οδήγησης USB COM και δεν χρειάζεται να υπάρχει εξωτερικός παράγοντας. Επομένως, στα Windows απαιτείται μόνο ένα αρχείο .inf .



Εικόνα 2.2 Μικροελεγκτής ATmega328

Ένα Arduino περιλαμβάνει ένα τμηματικό όργανο ελέγχου το οποίο επιτρέπει την απλή μορφή κειμένου δεδομένων που αποστέλλονται προς και από τη πλακέτα Arduino. Οι RX και TX λυχνίες LED στην πλακέτα θα αναβοσβήνουν όταν γίνεται μετάδοση δεδομένων μέσω του USB-to-chip σειριακή και USB σύνδεση με τον υπολογιστή (αλλά όχι για σειριακή επικοινωνία στις ακίδες 0 και 1).

## 2.8 Γλώσσα Προγραμματισμού

Το Arduino Uno μπορεί να προγραμματιστεί με τη γλώσσα Wiring (ουσιαστικά πρόκειται για τη C++ με κάποιες μετατροπές). Έρχεται με ένα φορτωτή εκκίνησης που μας επιτρέπει να ανεβάζουμε νέο κώδικα χωρίς τη χρήση εξωτερικού υλικού προγραμματιστή. Επικοινωνεί χρησιμοποιώντας το αρχικό πρωτόκολλο αναπτυξιακής κάρτας STK500.

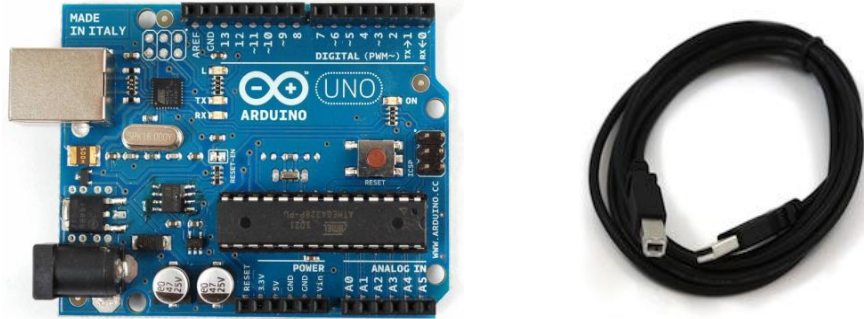
Το περιβάλλον ανάπτυξης του λογισμικού βασίζεται στην γλώσσα προγραμματισμού Processing και την γλώσσα προγραμματισμού Wiring, οι οποίες είναι ανοιχτού κώδικα (open source). Το περιβάλλον ανάπτυξης μπορεί κάποιος να το "κατεβάσει δωρεάν".

## 2.9 Εγκατάσταση του προγράμματος

Για να γίνει σωστή εγκατάσταση του προγράμματος, πρέπει να ακολουθηθεί μια σειρά από βήματα, ανάλογα με το λειτουργικό σύστημα που διαθέτει. Στην περίπτωση μας θα εγκατασταθεί σε λειτουργικό σύστημα των Windows XP.

### 1. Πλακέτα Arduino και καλώδιο USB

Στην παρούσα πτυχιακή εργασία επιλέχθηκε να χρησιμοποιηθεί η πλακέτα Arduino Uno. Θα χρειαστούμε ένα καλώδιο USB για να συνδεθούν πλακέτα και υπολογιστής.



### 2. Περιβάλλον Arduino

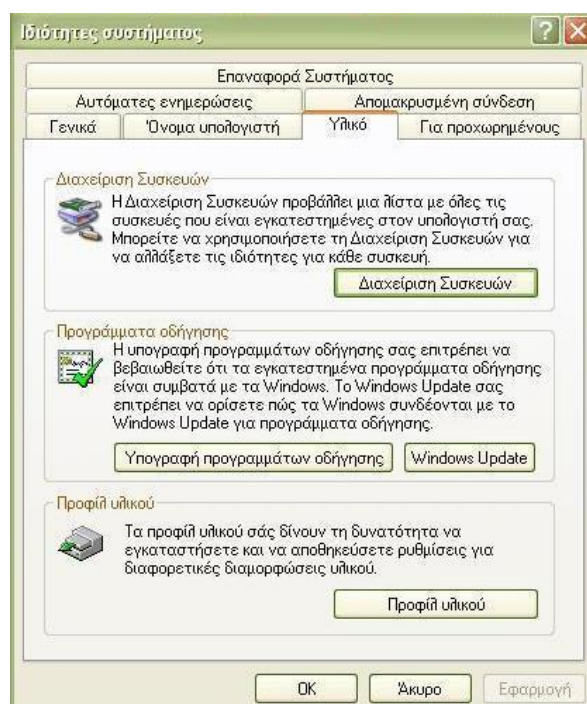
Μεταφορτώνουμε δωρεάν την τελευταία έκδοση Arduino-1.0 από την ιστοσελίδα <http://arduino.cc/en/Main/Software>.

### 3. Σύνδεση της πλακέτα στον υπολογιστή

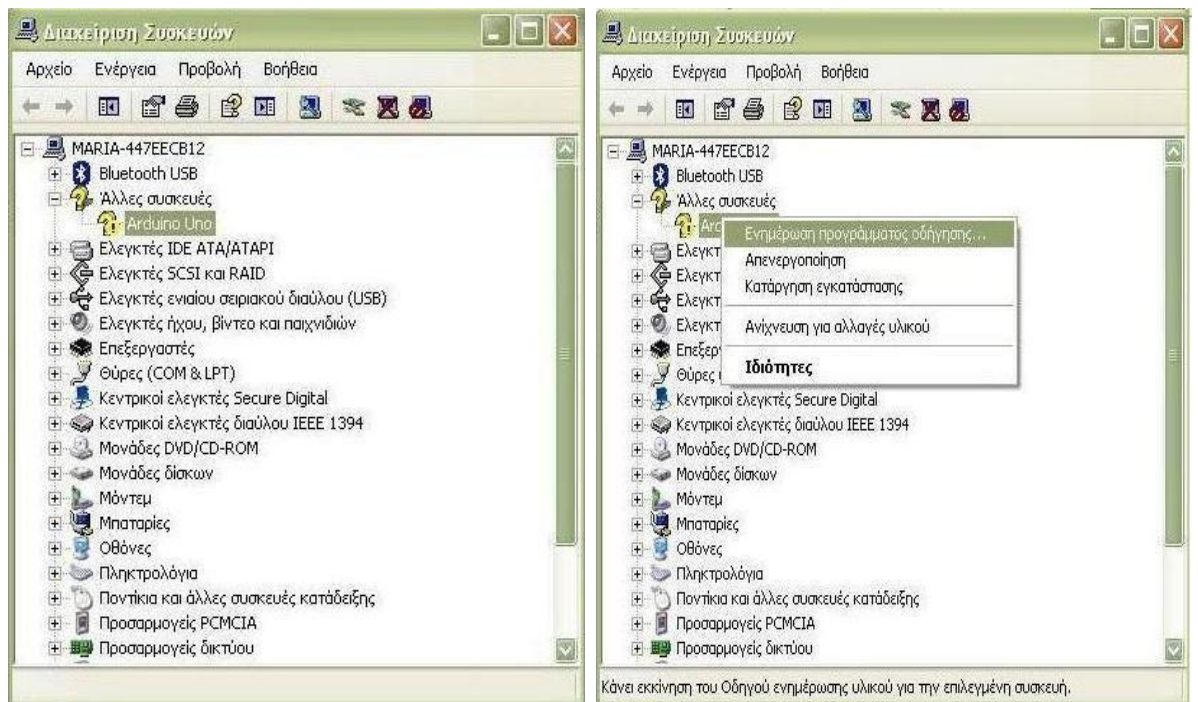
Συνδέουμε την πλακέτα Arduino Uno στον υπολογιστή χρησιμοποιώντας το καλώδιο USB. Παρατηρούμε ότι το LED της πλακέτας ανάβει.

### 4. Εγκατάσταση του προγράμματος

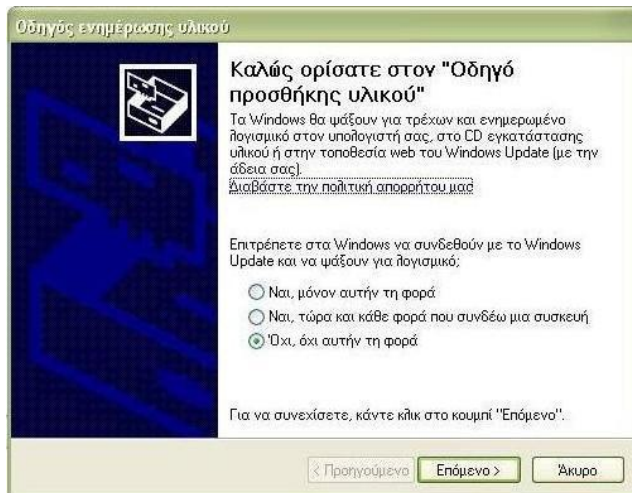
- + Κάνουμε κλικ στο μενού Έναρξη, και ανοίγουμε τον Πίνακα Ελέγχου.
- + Από τον Πίνακα Ελέγχου, μεταβαίνουμε στο «Σύστημα» και ακολούθως «Υλικό» και ανοίγουμε τη διαχείριση συσκευών.



- + Βλέπουμε στις συσκευές το όνομα Arduino Uno. Κάνουμε δεξί κλικ και επιλέγουμε το «Ενημέρωση προγράμματος οδήγησης».



- + Ξεκινάει εγκατάσταση λογισμικού για το Arduino.



- + Κάνουμε εγκατάσταση τα drivers στον υπολογιστή μας.



- + Περιμένουμε μέχρι να τελειώσει η εγκατάσταση λογισμικού για το Arduino.

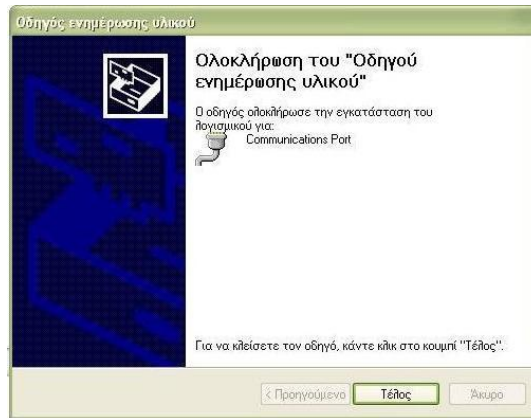


- + Τελιώνοντας παρατηρούμε ότι στις Θύρες (COM & LPT) εμφανίστηκε το Serial Port COM 15 για το Arduino που θα χρησιμοποιήσουμε. Οπότε το Arduino έχει προγραμματιστεί στη σειριακή θύρα 15.





## Κεφάλαιο 2 : Arduino Uno



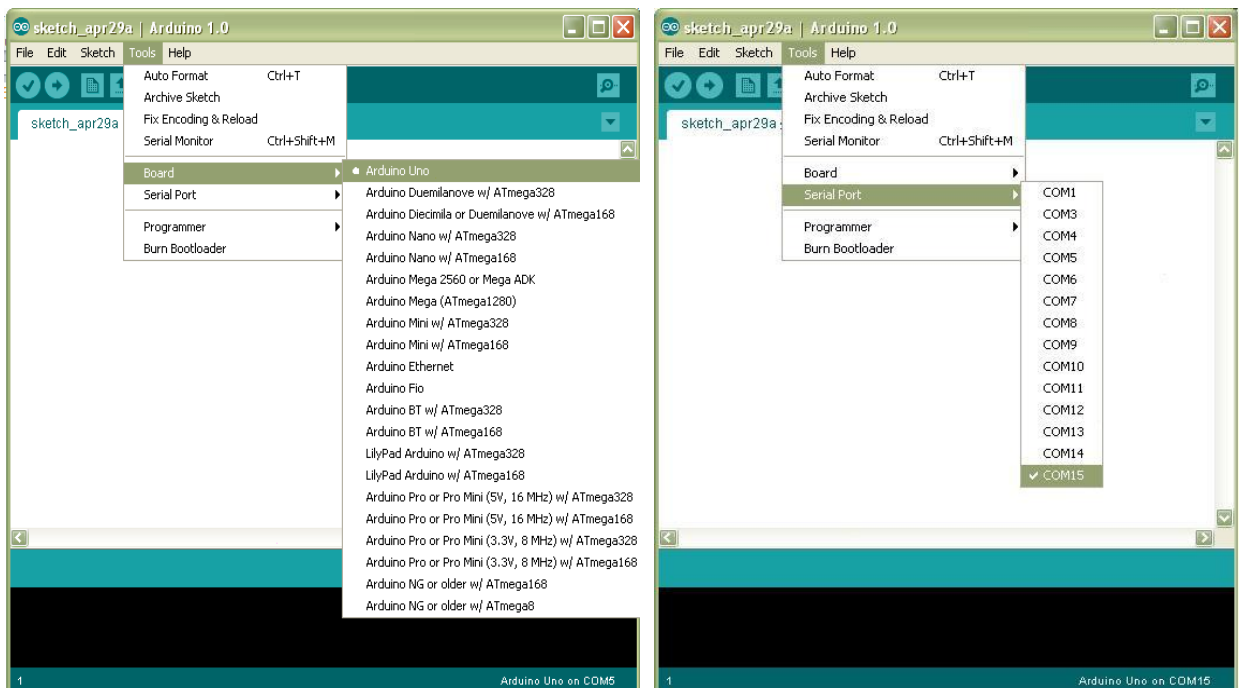
## 5. Έναρξη της εφαρμογής Arduino

Κάνουμε διπλό κλικ στην εφαρμογή Arduino.exe



## 6. Επιλογή Board και Σειριακής θύρας

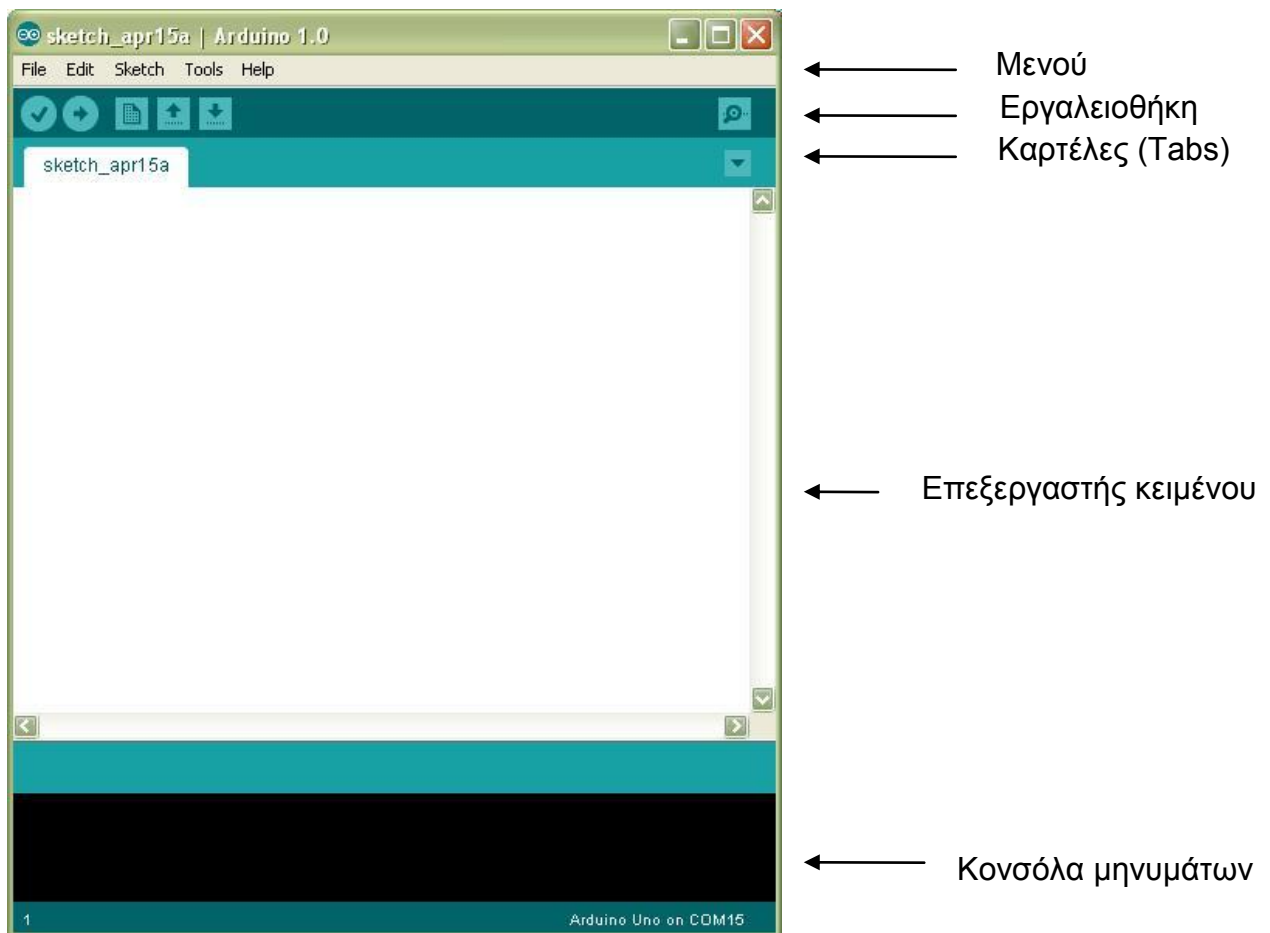
Από το περιβάλλον ανάπτυξης του, από το μενού Tools επιλέγουμε για Board το Arduino Uno και για Σειριακή θύρα το COM15.



## 2.10 Ολοκληρωμένο Περιβάλλον Ανάπτυξης του Arduino

Το περιβάλλον ανάπτυξης Arduino περιέχει μια περιοχή επεξεργασίας κειμένου για τη συγγραφή κώδικα, μια περιοχή μηνυμάτων, ένα μενού, μια γραμμή εργαλείων με κουμπιά για κοινές λειτουργίες, καθώς και μια σειρά από μενού. Συνδέεται με το υλικό Arduino για τη φόρτωση προγραμμάτων και για να επικοινωνούν μεταξύ τους.

Ένα ολοκληρωμένο πρόγραμμα συνήθως ονομάζεται sketch. Αυτό το sketch είναι γραμμένο με το πρόγραμμα επεξεργασίας κειμένου. Έχει δυνατότητες για την αντιγραφή/επικόλληση και για την αναζήτηση/αντικατάσταση κειμένου. Η κονσόλα απεικονίζει την έξοδο του κειμένου από το περιβάλλον Arduino συμπεριλαμβάνοντας πλήρη μηνύματα λάθους και άλλες πληροφορίες. Τα κουμπιά της γραμμής εργαλείων επιτρέπουν τον έλεγχο και το ανέβασμα των προγραμμάτων, τη δημιουργία νέου sketch, το άνοιγμα και την αποθήκευση των sketch και άνοιγμα της σειριακής οθόνης.



Εικόνα 2.3 Ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Arduino

Τα κουμπιά της γραμμής εργαλείων:



Verify/Compile (Έλεγχος/Μεταγλώττιση): Έλεγχος για λάθη στον κώδικα



Upload: Ανέβασμα του κώδικα στον μικροελεγκτή



New(Νέο): Δημιουργεί ένα νέο sketch



Open(Άνοιγμα): Παρουσιάζει ένα μενού με όλα τα sketch, κάνοντας κλικ σε ένα από αυτά θα ανοίξει μέσα στο τρέχον παράθυρο



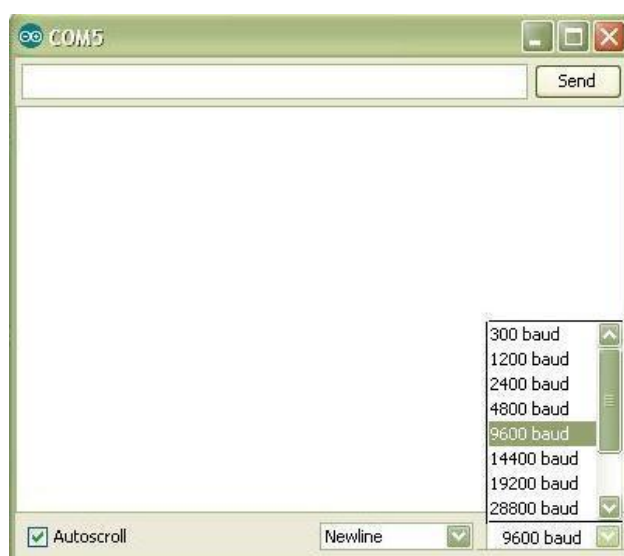
Save(Αποθήκευση): Αποθηκεύει το sketch



Serial Monitor(Σειριακή οθόνη): Ανοίγει την σειριακή οθόνη ώστε να μπορούμε να δώσουμε δεδομένα από το πληκτρολόγιο

## 2.11 Σειριακή οθόνη (Serial Monitor)

Εμφανίζει τα σειριακά δεδομένα που αποστέλλονται από την πλακέτα Arduino. Πιο συγκεκριμένα, η αποστολή δεδομένων στην πλακέτα γίνεται, εισάγοντας κείμενο και πατώντας το κουμπί send ή πατώντας το Enter. Επίσης, στο κάτω μέρος της σειριακής οθόνης, μπορεί να γίνει η επιλογή της κατάλληλης ταχύτητας (baud) από την λίστα που εμφανίζεται ανάλογα με την τιμή που θα επιλεγθεί στο προγραμματισμό του Arduino με το `Serial.begin()`.



Εικόνα 2.4: Serial Monitor

## 2.12 Η Δομή το προγράμματος

Ένα τυπικό πρόγραμμα Arduino έχει την παρακάτω δομή:

```
//δήλωση μεταβλητών
void setup ()
{
  //αρχικοποιήσεις
}
void loop ()
{
  //Κώδικας
}
```

Υπάρχουν δυο ειδικές συναρτήσεις που είναι μέρος του κάθε sketch του Arduino οι οποίες είναι η setup() και η loop(). Η setup() καλείται μια φορά, όταν το sketch ξεκινά ή όποτε κάνει επαναφορά (reset) η πλατφόρμα Arduino. Κυρίως, σε αυτήν γίνονται οι αρχικοποιήσεις των μεταβλητών, η ρύθμιση της κατάστασης των ακίδων (pins) και η προετοιμασία των βιβλιοθηκών. Αντιθέτως, η συνάρτηση loop() καλείται ξανά και ξανά επιτρέποντας έτσι στο πρόγραμμα να ανταποκριθεί σε εξωτερικά ερεθίσματα. Και οι δυο συναρτήσεις πρέπει να περιλαμβάνονται στο sketch, ακόμα και αν δεν περιέχουν κάτι και να είναι κενές.

## 2.13 Βασικές δομές και λειτουργίες προγραμματισμού

Παρακάτω, ακολουθούν μερικές από τις πιο βασικές δομές και λειτουργίες που μπορεί να αξιοποιηθεί ως εργαλεία κατά την συγγραφή ενός προγράμματος Arduino :

### Δομές ελέγχου ροής

- if (δομή ελέγχου μίας συνθήκης)
- if ... else (δομή ελέγχου πολλαπλών συνθηκών)
- for (δομή επαναληπτικού ελέγχου συνθήκης)
- while (δομή επαναληπτικού ελέγχου συνθήκης)
- do ... while (δομή επαναληπτικού ελέγχου συνθήκης)
- switch ... case (δομή ελέγχου περιπτώσεων)
- break (εντολή διακοπής μιας επαναληπτικής δομής)
- continue (εντολή παράλειψης της τρέχουσας επανάληψης)
- return (εντολή επιστροφής από μία συνάρτηση)
- goto (εντολή μετάβασης σε κάποιο σημείο του κώδικα)

### Αριθμητικοί τελεστές

- = (τελεστής εκχώρησης)
- + (τελεστής πρόσθεσης)
- - (τελεστής αφαίρεσης)
- \* (τελεστής πολλαπλασιασμού)

- / (τελεστής διαίρεσης)
- % (τελεστής υπόλοιπου ακεραίας διαίρεσης)

#### Λογικοί τελεστές

- && (λογική σύζευξη)
- || (λογική διάζευξη)
- ! (λογική άρνηση)

#### Δυαδικοί τελεστές

- & (δυαδική σύζευξη)
- | (δυαδική διάζευξη)
- ^ (δυαδική αποκλειστική διάζευξη)
- ~ (δυαδική άρνηση)
- << (δυαδική αριστερή ολίσθηση)
- >> (δυαδική δεξιά ολίσθηση)

#### Τελεστές αύξησης και μείωσης

- ++ (αύξηση κατά μία ακέραιη μονάδα)
- -- (μείωση κατά μία ακέραιη μονάδα)

#### Σύνθετοι τελεστές

- +=, -=, \*=, /=, %= (σύνθετοι αριθμητικοί τελεστές)
- &=, |=, ^=, ~=, <<=, >>= (σύνθετοι δυαδικοί τελεστές)

#### Τελεστές σύγκρισης

- == (ισότητα)
- != (ανισότητα)
- < (μικρότερο)
- > (μεγαλύτερο)
- <= (μικρότερο ή ίσο)
- >= (μεγαλύτερο ή ίσο)

#### Τελεστές δεικτών

- \* (τελεστής απόκτησης περιεχομένου)
- & (τελεστής απόκτησης διεύθυνσης)

#### Σταθερές

- HIGH (τιμή υψηλής στάθμης για μία επαφή εισόδου ή εξόδου)
- LOW (τιμή χαμηλής στάθμης για μία επαφή εισόδου ή εξόδου)
- false (λογικό επίπεδο ψεύδους σε μία συνθήκη)
- true (λογικό επίπεδο αλήθειας σε μία συνθήκη)
- INPUT (χρησιμοποιείται για τον ορισμό μίας επαφής ως είσοδο)
- OUTPUT (χρησιμοποιείται για τον ορισμό μίας επαφής ως έξοδο)
- A0, ..., A5 (συμβολοσταθερές για τις αναλογικές επαφές εισόδου)

### Τύποι δεδομένων

- `boolean` (λογική δυαδική τιμή)
- `char` (προσημασμένος χαρακτήρας 8 ψηφίων)
- `unsigned char` (μη προσημασμένος χαρακτήρας 8 ψηφίων)
- `byte` (μη προσημασμένος χαρακτήρας 8 ψηφίων)
- `int` (προσημασμένος ακέραιος αριθμός 16 ψηφίων)
- `unsigned int` (μη προσημασμένος ακέραιος αριθμός 16 ψηφίων)
- `word` (μη προσημασμένος ακέραιος αριθμός 16 ψηφίων)
- `long` (προσημασμένος ακέραιος αριθμός 32 ψηφίων)
- `unsigned long` (μη προσημασμένος ακέραιος αριθμός 32 ψηφίων)
- `float, double` (αριθμός κινητής υποδιαστολής απλής ακρίβειας)
- `String` (αντικείμενο αλφαριθμητικού με χρήσιμες μεθόδους)
- Ως αλφαριθμητικό μπορεί να θεωρηθεί και ο πίνακας χαρακτήρων

### Συναρτήσεις μετατροπής τύπων

- `char(), byte()`
- `int(), word(), long()`
- `float(), double()`

### Συναρτήσεις εισόδου και εξόδου

- `pinMode()` (ορίζει μια επαφή ως είσοδο ή έξοδο)

### Συναρτήσεις ψηφιακής εισόδου και εξόδου

- `digitalWrite()` (γράφει σε μία ψηφιακή επαφή εξόδου)
- `digitalRead()` (διαβάζει από μία ψηφιακή επαφή εισόδου)

### Συναρτήσεις αναλογικής εισόδου και εξόδου

- `analogReference()` (ορίζει την τάση αναλογικής αναφοράς)
- `analogWrite()` (γράφει PWM σήματα σε μία επαφή εξόδου)
- `analogRead()` (διαβάζει από μία αναλογική επαφή εισόδου)

### Προηγμένες συναρτήσεις εισόδου και εξόδου

- `tone()` (παράγει ένα τετραγωνικό σήμα ορισμένης συχνότητας)
- `noTone()` (διακόπτει την παραγωγή τετραγωνικών σημάτων)
- `shiftOut()` (ολισθαίνει τα ψηφία μιας τιμής σε μία επαφή εξόδου)
- `pulseIn()` (επιστρέφει την διάρκεια σε  $\mu\text{s}$  ενός παλμού HIGH ή LOW)

### Συναρτήσεις χρόνου

- `millis()` (διάρκεια εκτέλεσης του προγράμματος σε ms)
- `micros()` (διάρκεια εκτέλεσης του προγράμματος σε  $\mu\text{s}$ )
- `delay()` (παύση προγράμματος - η διάρκεια δίδεται σε ms)
- `delayMicroseconds()` (παύση προγράμματος - η διάρκεια δίδεται σε  $\mu\text{s}$ )

### Μαθηματικές και Τριγωνομετρικές συναρτήσεις

- `max()` (βρίσκει τον μεγαλύτερο ανάμεσα σε δύο αριθμούς)
- `min()` (βρίσκει τον μικρότερο ανάμεσα σε δύο αριθμούς)
- `abs()` (επιστρέφει την απόλυτη τιμή ενός αριθμού)
- `constrain()` (ελέγχει για υπερχείλιση ή υποχείλιση ορίων)
- `map()` (πραγματοποιεί γραμμικό μετασχηματισμό ορίων)
- `pow()` (επιστρέφει το αποτέλεσμα μίας δύναμης)

- `sqrt()` (επιστρέφει την ρίζα ενός αριθμού)
- `sin()` (υπολογίζει το ημίτονο ενός αριθμού)
- `cos()` (υπολογίζει το συνημίτονο ενός αριθμού)
- `tan()` (υπολογίζει την εφαπτομένη ενός αριθμού)

Συναρτήσεις γεννήτριας ψευδοτυχαίων αριθμών

- `random()` (δίδεται ένας νέος αριθμός από την γεννήτρια)
- `randomSeed()` (θέτει τον σπόρο της γεννήτριας παραγωγής)

Συναρτήσεις επεξεργασίας δυαδικών αριθμών

- `lowByte()` (επιστρέφει το δεξιότερο byte μίας μεταβλητής)
- `highByte()` (επιστρέφει το αριστερότερο byte μίας μεταβλητής)
- `bitRead()` (διαβάζει ένα συγκεκριμένο ψηφίο μίας μεταβλητής)
- `bitWrite()` (γράφει σε ένα συγκεκριμένο ψηφίο μιας μεταβλητής)
- `bitSet()` (γράφει την τιμή 1 σε κάποιο ψηφίο μίας μεταβλητής)
- `bitClear()` (γράφει την τιμή 0 σε κάποιο ψηφίο μιας μεταβλητής)
- `bit()` (υπολογίζει μία συγκεκριμένη δύναμη με βάση το 2)

Συναρτήσεις χρήσης ρουτινών εξυπηρέτησης διακοπών

- `attachInterrupt()` (ενεργοποιεί μία ρουτίνα εξυπηρέτησης διακοπής)
- `detachInterrupt()` (απενεργοποιεί μία ρουτίνα εξυπηρέτησης διακοπής)

Συναρτήσεις ενεργοποίησης και απενεργοποίησης διακοπών

- `interrupts()` (ενεργοποιεί τα σήματα διακοπής)
- `noInterrupts()` (απενεργοποιεί τα σήματα διακοπής)

Υποστήριξη σειριακής επικοινωνίας

- `Serial` (αντικείμενο σειριακής επικοινωνίας με χρήσιμες μεθόδους)

## 2.14 Ψηφιακές ακίδες (Digital pins)

Οι ακίδες αυτές στο Arduino μπορούν να ρυθμιστούν είτε ως είσοδοι είτε ως έξοδοι, όμως από προεπιλογή είναι ρυθμισμένες ως είσοδοι. Επίσης αξίζει να σημειωθεί, ότι η πλειοψηφία των αναλογικών ακίδων του Arduino (Atmega), μπορεί να ρυθμιστεί και να χρησιμοποιηθεί, με τον ίδιο ακριβώς τρόπο όπως οι ψηφιακές ακίδες. Οι συναρτήσεις ψηφιακής εισόδου και εξόδου είναι οι παρακάτω:

- `pinMode()`: Ρυθμίζει τη συγκεκριμένη ακίδα να συμπεριφέρεται ως είσοδος/έξοδος.

**Σύνταξη:** `pinMode(pin, mode)`

**Παράμετροι:**

`pin`: Ο αριθμός της ακίδας της οποίας η λειτουργία είναι επιθυμητό να αλλάξει.

`mode`: INPUT/OUTPUT

- digitalWrite(): Γράφει μια υψηλή (HIGH) ή μια χαμηλή (LOW) τιμή σε μια ψηφιακή ακίδα. Αν η ακίδα έχει ρυθμιστεί ως έξοδος με την συνάρτηση pinMode(), τότε η τάση της θα καθορίσει στην αντίστοιχη τιμή: 5V για HIGH και 0V για LOW. Αν η ακίδα έχει ρυθμιστεί ως είσοδος, γράφοντας HIGH στην συνάρτηση digitalWrite() θα ενεργοποιήσει μια εσωτερική pullup-αντίσταση των 20 K ενώ γράφοντας LOW θα την απενεργοποιήσει.

**Σύνταξη:** digitalWrite(pin,value)

**Παράμετροι:**

pin: Ο αριθμός της ακίδας της οποίας η λειτουργία είναι επιθυμητό να αλλάξει.

Value: INPUT/OUTPUT

- digitalRead(): Διαβάζει την τιμή από μια συγκεκριμένη ψηφιακή ακίδα, που είναι είτε HIGH είτε LOW.

**Σύνταξη:** digitalRead(pin)

**Παράμετροι:**

pin: Ο αριθμός της ακίδας της οποίας η λειτουργία είναι επιθυμητό να αλλάξει.

**Επιστρέφει:** HIGH/LOW

## 2.15 Αναλογικές ακίδες εισόδου (Analog input pins)

Οι ελεγκτές Atmega που χρησιμοποιούνται για την πλατφόρμα Arduino περιέχουν έναν ενσωματωμένο αναλογικό-σε-ψηφιακό μετατροπέα 6 καναλιών. Ο μετατροπέας διαθέτει ανάλυση 10 bit, επιστρέφοντας ακέραιους από 0 έως 1023. Ενώ η κύρια λειτουργία της αναλογικής ακίδας για τους περισσότερους χρήστες Arduino είναι να διαβάζει αναλογικούς αισθητήρες, οι αναλογικές ακίδες έχουν επίσης όλες τις λειτουργίες των γενικών ακίδων εισόδου/εξόδου. Οι συναρτήσεις αναλογικής εισόδου και εξόδου είναι οι παρακάτω:

- analogWrite(): Γράφει μια αναλογική τιμή (PWM κύμα) σε μια ακίδα. Μπορεί να χρησιμοποιηθεί για παράδειγμα να ανάψει ένα LED σε διάφορες φωτεινότητες ή να οδηγήσει ένα κινητήρα σε διάφορες ταχύτητες. Μετά από μια κλήση της analogWrite(), η ακίδα θα δημιουργήσει ένα σταθερό τετραγωνικό κύμα του καθορισμένου κύκλου λειτουργίας μέχρι την επόμενη κλήση της analogWrite() (ή μια κλήση της digitalWrite() ή digitalRead() για την ίδια ακίδα). Η συχνότητα του σήματος PWM είναι περίπου 490 Hz. Στις περισσότερες πλατφόρμες Arduino η συνάρτηση αυτή λειτουργεί στις ακίδες 3, 5, 6, 9, 10, 11.

**Σύνταξη:** analogWrite(pin, value)

**Παράμετροι:**

pin: Ο αριθμός της ακίδας της οποίας θα γράψει επάνω

value: ο κύκλος λειτουργίας μεταξύ 0 και 255



- analogRead(): Διαβάζει την τιμή από την καθορισμένη αναλογική ακίδα.  
Σύνταξη: analogRead(pin)  
Παράμετροι:  
pin: Ο αριθμός της αναλογικής ακίδας εισόδου από όπου θα διαβάζει  
Επιστέφει: ακέραιο από 0 έως 1023

## 2.16 Υποστήριξη Βιβλιοθηκών (Libraries)

Η χρήση βιβλιοθηκών προσφέρουν περισσότερο λειτουργικότητα σε συνεργασία με το υλικό και τον χειρισμό των δεδομένων. Για να χρησιμοποιηθεί μια βιβλιοθήκη σε ένα sketch, μπορεί να επιλεγεί από το μενού *Sketch* → *Import Library*. Αυτό θα εισάγει μια ή περισσότερες βιβλιοθήκες #include δηλώσεις στην κορυφή του sketch. Επειδή οι βιβλιοθήκες φορτώνονται στην πλακέτα με το sketch, αυξάνουν το μέγεθος του χώρου που καταλαμβάνεται. Εάν ένα sketch δεν χρειάζεται πλέον μια βιβλιοθήκη, απλά μπορούμε να την διαγράψουμε από την κορυφή του κώδικα.

Για την εγκατάσταση των βιβλιοθηκών που δεν υπάρχουν ήδη στο λογισμικό, μπορεί να δημιουργηθεί ένας κατάλογος με την ονομασία libraries (βιβλιοθήκες), μέσα στον κατάλογο του sketchbook. Στην συνέχεια αποσυμπιέζουμε τη βιβλιοθήκη εκεί.

Παρακάτω ακολουθούν μερικές από τις βιβλιοθήκες που υποστηρίζονται από το Arduino.

Επικοινωνίας (δικτύωση και πρωτόκολλα):

- Messenger - για την επεξεργασία κειμένου με βάση τα μηνύματα από τον υπολογιστή.
- NewSoftSerial - βελτιωμένη έκδοση της βιβλιοθήκης SoftwareSerial.
- OneWire - συσκευές ελέγχου (της Dallas Semiconductor) που χρησιμοποιούν το πρωτόκολλο one Wire.
- PS2Keyboard - διαβάζει χαρακτήρες από ένα πληκτρολόγιο PS2.
- Simple Message System - στέλνει μηνύματα μεταξύ Arduino και του υπολογιστή
- SSerial2Mobile - αποστολή μηνυμάτων κειμένου ή e-mail χρησιμοποιώντας ένα κινητό τηλέφωνο (μέσω εντολών AT μέσω σειράς λογισμικού)
- Webduino - επεκτάσιμη βιβλιοθήκη web server (για χρήση με το Arduino Ethernet Shield)
- X10 - Αποστολή σημάτων X10 μέσω γραμμών εναλλασσόμενου ρεύματος
- Xbee - για την επικοινωνία με XBees σε λειτουργία API
- SerialControl - Τηλεχειριστήριο άλλες Arduino πάνω από μια σειριακή σύνδεση
- Servo – για τον έλεγχο κινητήρων τύπου Servo.

Ανίχνευσης:

- Capacitive Sensing - δύο ή περισσότερες ακίδες σε αισθητήρες πυκνωτή
- Debounce - για την ανάγνωση θορυβώδη ψηφιακών εισόδων (π.χ. από τα κουμπιά).

#### Εμφάνιση και LED:

- Improved LCD library - διορθώνει σφάλματα αρχικοποίησης LCD στην επίσημη Arduino LCD βιβλιοθήκη.
- GLCD – γραφικές ρουτίνες για LCD με βάση την KS0108 ή ισοδύναμο chipset.
- LedControl - για τον έλεγχο των LED ή επτά τμημάτων οθόνες με MAX7221 ή MAX7219.
- LedControl - μια εναλλακτική λύση στη βιβλιοθήκη Matrix για την οδήγηση με πολλαπλούς LED.
- LedDisplay - τον έλεγχο της HCMS-29xx οθόνη LED.

#### Συχνότητα παραγωγής ήχου:

- Tone - αναπαράγει κύματα ήχου συχνότητας στο παρασκήνιο σε κάθε καρφίτσα του μικροελεγκτή.

#### Κινητήρες και PWM:

- TLC5940 - 16 κανάλι 12 bit PWM ελεγκτή.

#### Χρονοδιάγραμμα:

- DateTime - μια βιβλιοθήκη για την παρακολούθηση της τρέχουσας ημερομηνίας και ώρας.
- MsTime2 - χρησιμοποιεί το χρονόμετρο διακοπής 2 για να ενεργοποιήσει μια δράση κάθε χιλιοστά του δευτερολέπτου N.

#### Βοηθητικά προγράμματα:

- Streaming - μια μέθοδο για την απλοποίηση δηλώσεων εκτύπωσης.

*Για την συγγραφή των παραπάνω υποκεφαλαίων πολύ χρήσιμες ήταν οι ιστοσελίδες [1], [2], [3], [4] και [5].*

# ΚΕΦΑΛΑΙΟ 8

# Processing



Η Processing είναι μια ανοιχτού κώδικα γλώσσα προγραμματισμού και παρέχει περιβάλλον ανάπτυξης για άτομα που θέλουν να δημιουργήσουν εικόνες, κινούμενα σχέδια και διάφορες αλληλεπιδράσεις.

Αρχικά, αναπτύχθηκε ως ένα σχεδιαστικό πρόγραμμα για να διδάξει βασικές αρχές προγραμματισμού μέσα σε ένα οπτικό πλαίσιο, όμως στη συνέχεια εξελίχθηκε σε ένα εργαλείο δημιουργίας ολοκληρωμένων επαγγελματικών εργασιών. Αυτή τη στιγμή υπάρχουν δεκάδες χιλιάδες σπουδαστές, καλλιτέχνες, σχεδιαστές, ερευνητές και χομπιστές που χρησιμοποιούν την Processing για διδασκαλία, προτυποποίηση και παραγωγή.

## 8.1 Χαρακτηριστικά της Processing

- Είναι ελεύθερο/ανοιχτό λογισμικό με άδεια χρήσης GPL/LGPL.
- Είναι πολυπλατφορμική, μπορεί να τρέξει σε λειτουργικά συστήματα GNU/Linux, Mac OS X και Windows.
- Δημιουργεί διαδραστικά προγράμματα, χρησιμοποιώντας δισδιάστατα(2D) ή τρισδιάστατα (3D) γραφικά.
- Ενσωμάτωση της OpenGL για επιτάχυνση 3D.
- Δημιουργία stand-alone desktop εφαρμογών και Web-based εφαρμογών (applets).
- Υπάρχουν αρκετές βιβλιοθήκες επέκτασης της γλώσσας για εφαρμογές ήχου, βίντεο, τεχνητής όρασης, κ.α.

Η Processing βασίστηκε στις δυνατότητες γραφικών της γλώσσας προγραμματισμού Java, απλοποιώντας τη χρήση και δημιουργώντας νέα χαρακτηριστικά.

## 8.2 Έργα βασισμένα στην Processing

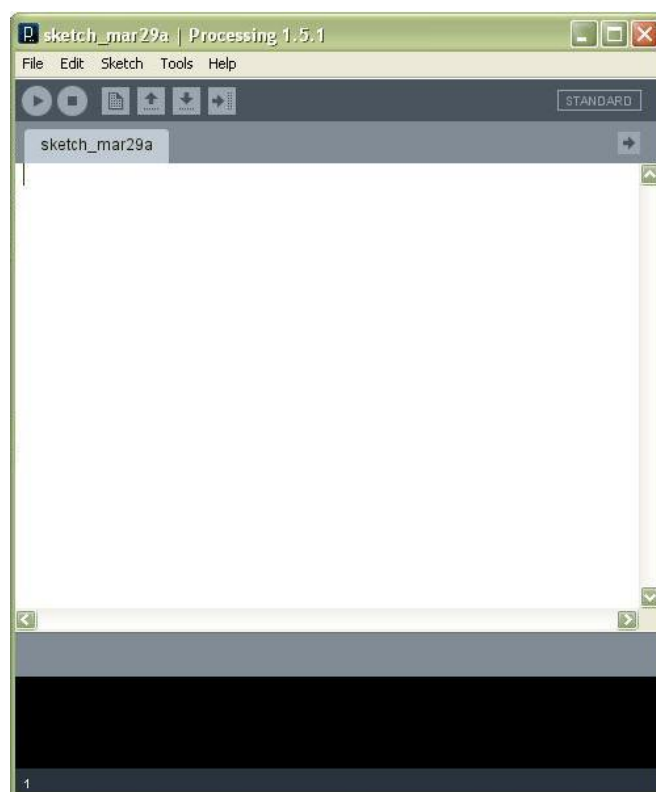
Η Processing είναι πηγή έμπνευσης για αρκετά έργα ανοιχτού κώδικα. Δύο από τα σημαντικότερα έργα που χρησιμοποιούν τη γλώσσα και το περιβάλλον της είναι η

Wiring και το Arduino, οι οποίες στοχεύουν στον απλό και άμεσο προγραμματισμό hardware. Σε αντίθεση με τη Wiring, το Arduino δημιούργησε πλακέτες απλές, οικονομικές και ευκολότερες στη χρήση, ενώ κατ' επέκταση χρησιμοποιεί μια παραλλαγή της γλώσσας Processing για τον προγραμματισμό μικροελεγκτών AVR της εταιρείας Atmel. Άλλα έργα που εμπνεύστηκαν από την Processing είναι τα: Design By Numbers, Fritzing, Mobile Processing, Processing.js, Spde, Processing in Clojure και Processing Monsters.

### 8.3 Η δομή του προγράμματος

Ένα τυπικό πρόγραμμα Processing έχει την παρακάτω δομή:

```
//δήλωση μεταβλητών
void setup()
{
  //αρχικοποιήσεις
}
void draw()
{
  //Κώδικας
}
```



- ← Μενού
- ← Εργαλειοθήκη
- ← Καρτέλες (Tabs)
  
- ← Επεξεργαστής κειμένου
  
- ← Μηνύματα κονσόλα

Εικόνα 9.1: Ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) του Processing

Όπως παρατηρούμε, υπάρχουν δύο βασικές συναρτήσεις σε ένα τυπικό πρόγραμμα της Processing. Η συνάρτηση `setup()` εκτελείται στην αρχή του προγράμματος και για μια φορά. Χρησιμοποιείται για τις αρχικοποιήσεις ιδιοτήτων και βιβλιοθηκών, για παράδειγμα το μέγεθος της οθόνης, το χρώμα του φόντου, το φόρτωμα εικόνων κ.α. Οι μεταβλητές που δηλώνονται στην `setup()` δεν είναι προσβάσιμες από άλλες συναρτήσεις, συμπεριλαμβανομένης και της `draw()` που θα αναλυθεί αμέσως μετά. Η συνάρτηση `draw()` εκτελείται μετά από την `setup()` και ο κώδικας που γράφεται μέσα στην συνάρτηση αυτή επαναλαμβάνεται συνεχώς έως ότου τερματιστεί η εφαρμογή ή μέχρι να κληθεί η συνάρτηση `noLoop()`.

## 8.4 Βιβλιοθήκες και εργαλεία

Η Processing συνοδεύεται από μια πληθώρα βιβλιοθηκών και εργαλείων ανοικτού κώδικα, που ως στόχο έχουν να επεκτείνουν τις δυνατότητες της γλώσσας. Υπάρχουν βιβλιοθήκες για κάθε σκοπό – όπως για δημιουργία τρισδιάστατων γραφικών, δημιουργία κινουμένων σχεδίων, τεχνητή όραση, μαθηματικά, επεξεργασία ήχου, δημιουργία προσομοιώσεων κ.α. Μπορούμε να κατεβάσουμε αρκετές βιβλιοθήκες από την διεύθυνση <http://processing.org/reference/libraries>. Επίσης, υπάρχουν εργαλεία τα οποία μπορούμε να τα βρούμε στο μενού «Tools», τα οποία μας βοηθούν σε πολλές εργασίες.

Πρέπει να φορτωθούν οι βιβλιοθήκες `Arduinoscope` από την ιστοσελίδα <http://code.google.com/p/arduinoscope/downloads/list>, και `ControlP5` από την ιστοσελίδα <http://www.sojamo.de/libraries/controlP5/>. Αφού τα κατεβάσουμε πρέπει να κάνουμε αντίγραφο και των δυο βιβλιοθηκών και να επικολλήσουμε στον φάκελο `processing-1.5\modes\java\libraries` με ονόματα `ControlP5` και `arduinoscope` αντίστοιχα (τα ονόματα έχουν μεγάλη σημασία).

*Για την συγγραφή αυτού του κεφαλαίου πολύ χρήσιμα ήταν οι ιστοσελίδες [22], [23].*

## ΚΕΦΑΛΑΙΟ 9

## Βιβλιογραφία – Πηγές πληροφοριών

Βιβλία και σημειώσεις που χρησιμοποιήθηκαν:

- [A] Δρ. Βολογιαννίδης Σταύρος (2009). Ευφυής Έλεγχος, Θεωρία και Εφαρμογής.  
[B] Banzi, M. (2009). *Getting Started with Arduino*. O'Reilly.

Ιστοσελίδες που χρησιμοποιήθηκαν:

*Arduino*

[1] <http://arduino.cc/en/Guide/Environment?from=Tutorial.Bootloader>

[2] <http://arduino.cc/en/Guide/Windows>

[3] <http://arduino.cc/en/Reference/HomePage>

[4] <http://arduino.cc/en/Main/ArduinoBoardUno>

[5] <http://arduino.cc/en/Tutorial/Memory>

*Processing*

[22] <http://www.arduino.cc/playground/Interfacing/Processing>

[23] <http://www.processing.org/reference/>

## ΚΕΦΑΛΑΙΟ 3

### Πρακτικό μέρος

- Επίσκεψη και περιήγηση στην Ιστοσελίδα [www.arduino.cc](http://www.arduino.cc)
- Κατέβασμα και εγκατάσταση εφαρμογής ARDUINO IDE
- Επικοινωνία εφαρμογής ARDUINO IDE και της πλακέτας ARDUINO UNO

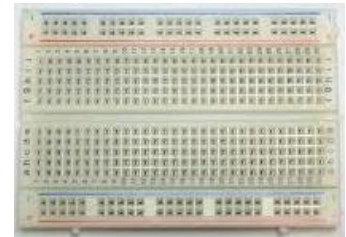
### 3.1 Απαραίτητα Υλικά

Τα απαραίτητα υλικά που θα χρειαστούμε στη συνέχεια για την ολοκλήρωση του πρακτικού μέρους είναι τα παρακάτω:

❖ Arduino Uno



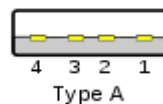
❖ BreadBoard



❖ Καλώδια



❖ USB καλώδια πλατφόρμων Arduino



❖ Διάφορα

εξαρτήματα : Αντιστάσεις, LED, Ημιαγωγοί κλπ



## 3.2 Εφαρμογή 1<sup>η</sup>

---

**Συνεχές αναβόσβημα ενός Led με περίοδο 1sec, με αλλαγή της στάθμης στην ψηφιακή θύρα I/O του μικροελεγκτή. (Οι γραμμές με πορτοκαλί είναι κώδικας-εντολές όλα τα άλλα είναι σχόλια)**

Το Arduino αποτελείται από δεκατρία ψηφιακά pin, τα οποία μπορούμε να τα χρησιμοποιήσουμε το κάθε ένα ξεχωριστά, είτε για είσοδο είτε για έξοδο. Μπορούμε να τα προγραμματίσουμε να συμπεριφέρονται όπως εμείς θέλουμε, αρκεί να κάνουμε τις σωστές δηλώσεις στο κώδικα που θα φορτώσουμε στη πλακέτα.

Η έξοδος του κάθε pin μπορεί να προγραμματιστεί να δίνει τιμές HIGH ή LOW. Λέγοντας HIGH ενώνουμε το δυαδικό '1' και έχουμε τάση εξόδου 5V DC, ενώ το LOW είναι το δυαδικό '0' και έχει τάση εξόδου 0V DC (ground).

Παρακάτω θα δούμε ένα παράδειγμα για να μπορέσουμε να κατανοήσουμε τον προγραμματισμό των ψηφιακών pin του Arduino. Θα ενώσουμε ένα led στη board και θα το προγραμματίσουμε να ανάβει και να σβήνει σε χρονικά διαστήματα ενός δευτερολέπτου

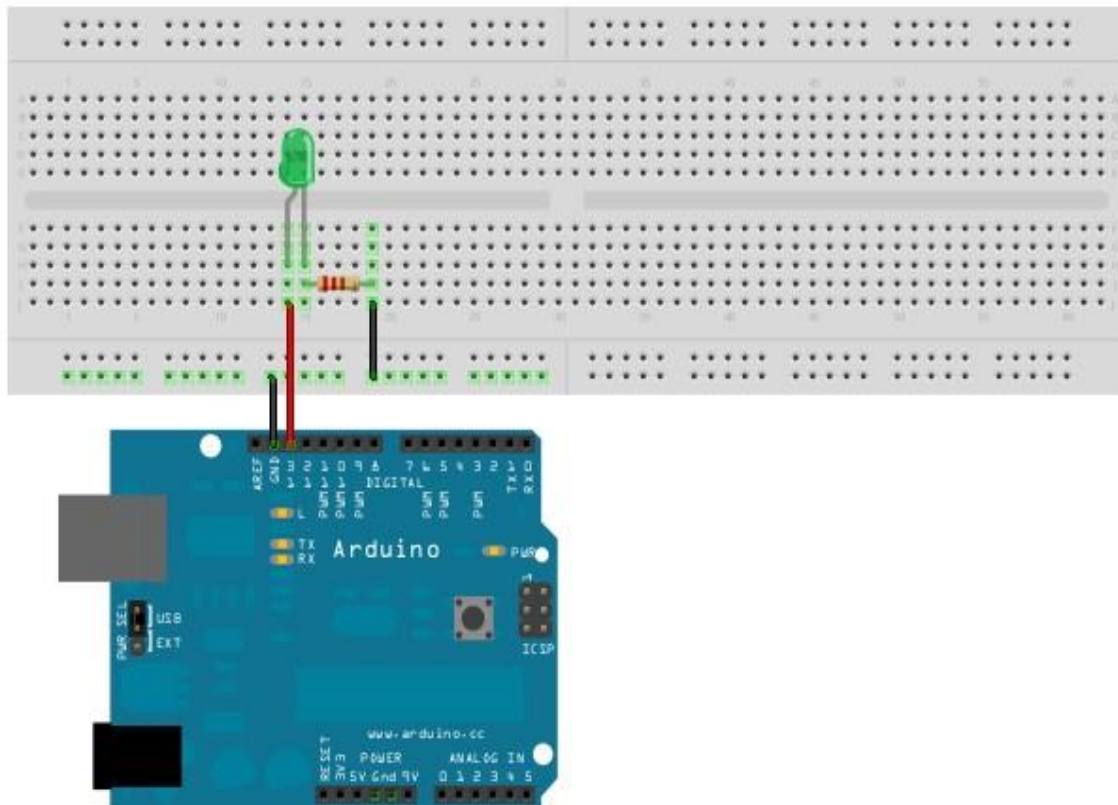
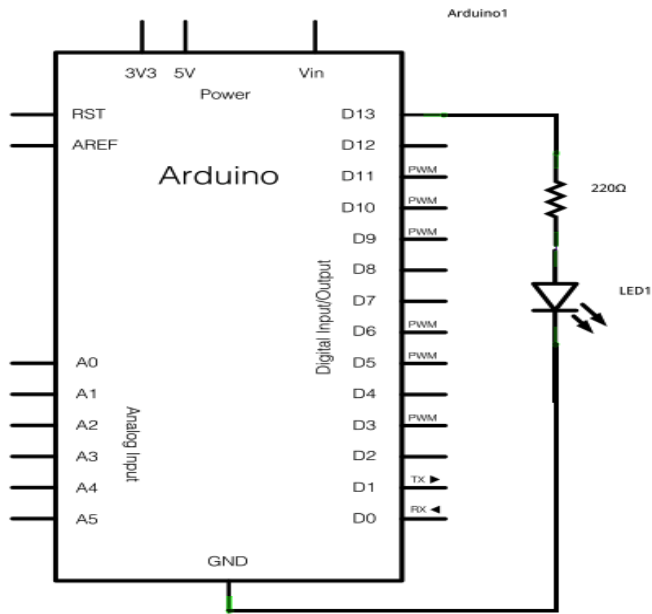
### *Blink*

Αυτό το παράδειγμα παρουσιάζει το απλούστερο πράγμα που μπορείτε να κάνετε με ένα Arduino για να δείτε τη φυσική έξοδο: αναβοσβήνει ένα LED.

### **Απαραίτητα Υλικά**

- Arduino Board
- breadboard
- LED, αντίσταση 220Ω,

**Κύκλωμα** Για να υλοποιήσετε το κύκλωμα, συνδέστε μια αντίσταση 220Ω με το pin 13. Κατόπιν συνδέστε τον μακρύτερο ακροδέκτη (άνοδο) με την αντίσταση. Συνδέστε το κοντό ακροδέκτη (κάθοδο) με τη γείωση. Κατόπιν συνδέστε το Arduino με τον υπολογιστή σας, ξεκινήστε το πρόγραμμα Arduino, και πληκτρολογήστε τον παρακάτω κώδικα. Τα περισσότερα Arduino ήδη έχουν ένα LED στο pin 13. Εάν τρέχετε αυτό το παράδειγμα χωρίς εξωτερικό LED και αντίσταση, πρέπει να δείτε το ενσωματωμένο LED να αναβοσβήνει.



Για την δημιουργία της εικόνας χρησιμοποιήθηκε το πρόγραμμα [Fritzing](#). Για περισσότερα παραδείγματα κυκλωμάτων, δείτε: [Fritzing project page](#)

## Κώδικας

```
// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13; // Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα led και της δίνουμε την
τιμή 13 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα
συνδέσουμε το led.

// the setup routine runs once when you press reset:

void setup() { // Η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
// initialize the digital pin as an output.
pinMode(led, OUTPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή led, ότι το pin 13 θα
λειτουργεί σαν ψηφιακή έξοδος 0-5V
}

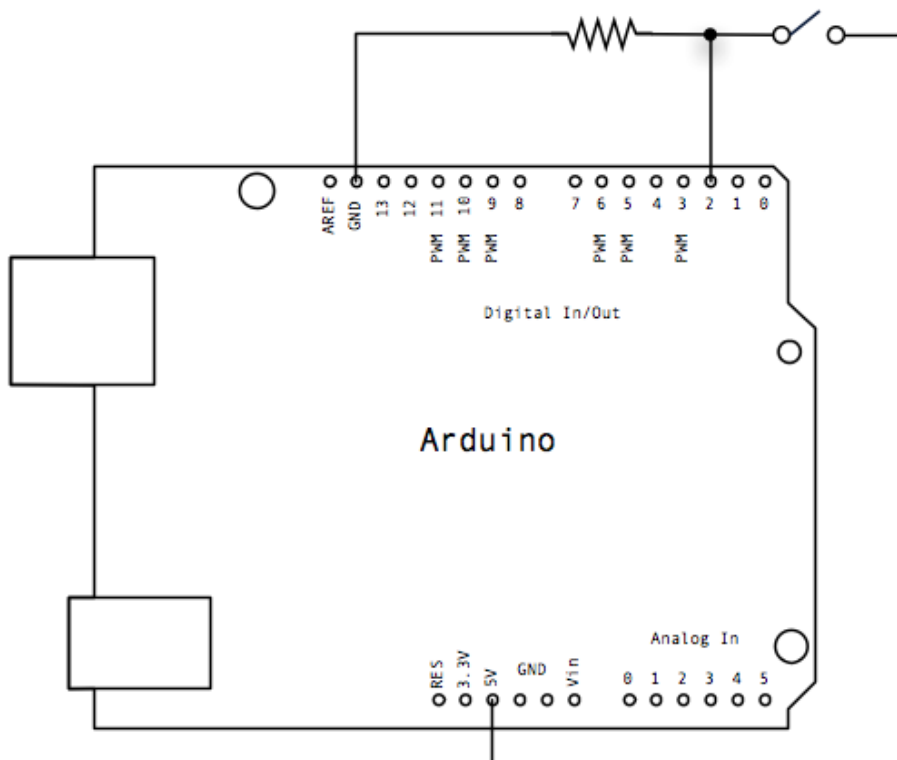
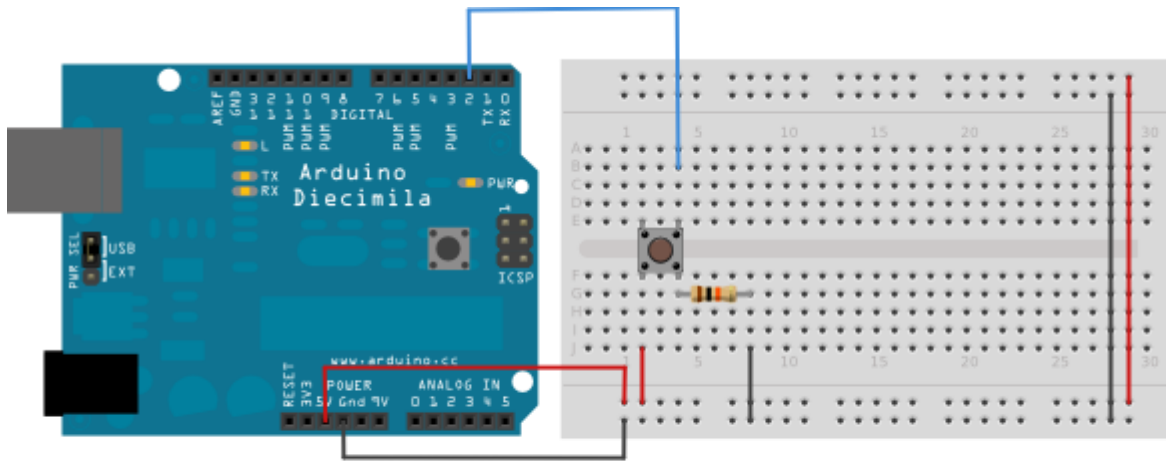
// the loop routine runs over and over again forever:

void loop() { // Η κύρια ρουτίνα του προγράμματος που εκτελείται συνεχώς
digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
// Βγάλε λογικό 1 (5V) στο pin 13, δηλαδή άναψε το led
delay(1000); // wait for a second
// Συνάρτηση καθυστέρησης χρόνου 1000ms
digitalWrite(led, LOW); // turn the LED off by making the voltage LOW
//Βγάλε λογικό 0 (0V) στο pin 13, δηλαδή σβήσε το led
delay(1000); // wait for a second
// Συνάρτηση καθυστέρησης χρόνου 1000ms
} // Επιστρέφει στην γραμμή digitalWrite(led, HIGH);
```

[\[Get Code\]](#)

### 3.3 Εφαρμογή 2<sup>η</sup>

Άναμμα και σβήσιμο Led ελεγχόμενο από εξωτερικό button με την διαδικασία polling



```

// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2; // the number of the pushbutton pin
// Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με όνομα buttonPin και της δίνουμε την τιμή 2 με
// σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε το
// button.
const int ledPin = 13; // the number of the LED pin
// Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με όνομα ledPin και της δίνουμε την τιμή 13 με
// σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε το
// led.
// variables will change:
int buttonState = 0; // variable for reading the pushbutton status
// Ορίζουμε μία μεταβλητή τύπου ? Integer (Ακεραίου) με όνομα buttonState και της δίνουμε αρχικά
// την τιμή 0 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα για να αποθηκεύουμε την
// κατάσταση λειτουργίας του button (on-off)
void setup() { // Η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
// initialize the LED pin as an output:
pinMode(ledPin, OUTPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή ledPin, ότι το pin 13 θα
// λειτουργεί σαν ψηφιακή έξοδος 0-5V
// initialize the pushbutton pin as an input:
pinMode(buttonPin, INPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή buttonPin, ότι το pin 2
// θα λειτουργεί σαν ψηφιακή είσοδος 0-5V
}
void loop() { // Η κύρια ρουτίνα του προγράμματος που εκτελείται συνεχώς
// read the state of the pushbutton value:
buttonState = digitalRead(buttonPin); // Διαβάζουμε την ψηφιακή τιμή που έχει το Pin 2 (εκεί που
// συνδέεται το button) και την αποθηκεύουμε στην μεταβλητή buttonState
// check if the pushbutton is pressed.
// if it is, the buttonState is HIGH:
if (buttonState == HIGH) { //Ελέγχουμε αν το button είναι πατημένο, αν ναι τότε:
// turn LED on:
digitalWrite(ledPin, HIGH); // Βγάλε λογικό 1 (5V) στο pin 13, δηλαδή άναψε το led
}
else { //αλλιώς:
// turn LED off:
digitalWrite(ledPin, LOW); //Βγάλε λογικό 0 (0V) στο pin 13, δηλαδή σβήσε το led
}
}
}

```

## 3.4 Εφαρμογή 3<sup>η</sup>

**Άναμμα και σβήσιμο Led ελεγχόμενο από εξωτερικό button με την διαδικασία Interrupt**  
(Η διαφορά με την προηγούμενη περίπτωση είναι ότι πριν η ενεργοποίηση γίνεται μετά το loop ενώ τώρα άμεσα με την ενεργοποίηση της διακοπής)

```

const byte LED = 13; // Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με όνομα LED και της
δίνουμε την τιμή 13 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που
θα συνδέσουμε το led.
const byte BUTTON = 2; // Ορίζουμε μία σταθερά τύπου Integer (Ακεραίου) με όνομα BUTTON και
της δίνουμε την τιμή 2 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε
που θα συνδέσουμε το button.

// Interrupt Service Routine (ISR)
void pinChange () // Δημιουργία ρουτίνας εξυπηρέτησης διακοπής με όνομα pinChange (δηλώνεται
μετά τις μεταβλητές και πριν την ρουτίνα αρχικών ρυθμίσεων του μικροελεγκτή

{
  noInterrupts (); // Απενεργοποίηση των διακοπών με σκοπό να μην συμβεί κάποια άλλη διακοπή για
όσο χρόνο εξυπηρετείται η παρούσα διακοπή
  if (digitalRead (BUTTON) == HIGH) // Διαβάζουμε την ψηφιακή τιμή που έχει το Pin 2 (εκεί που
συνδέεται το button) και ελέγχουμε αν το button είναι πατημένο, αν ναι τότε
    digitalWrite (LED, HIGH); // Βγάλε λογικό 1 (5V) στο pin 13, δηλαδή άναψε το led
  else //αλλιώς:
    digitalWrite (LED, LOW); //Βγάλε λογικό 0 (0V) στο pin 13, δηλαδή σβήσε το led
  interrupts (); //Ενεργοποίηση των διακοπών
} // end of pinChange

void setup () // Η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
{
  pinMode (LED, OUTPUT); // so we can update the LED
                          // Ορίζουμε χρησιμοποιώντας την μεταβλητή LED, ότι το pin 13 θα
λειτουργεί σαν ψηφιακή έξοδος 0-5V
  digitalWrite (BUTTON, HIGH); // internal pull-up resistor
                          // Ορίζουμε χρησιμοποιώντας την μεταβλητή BUTTON, ότι το pin 2
θα λειτουργεί σαν ψηφιακή είσοδος 0-5V

  attachInterrupt (0, pinChange, CHANGE); // attach interrupt handler
                          // Ενεργοποίηση της ρουτίνας που θα εξυπηρετεί την
διακοπή. Η πρώτη παράμετρος με τιμή 0 μας ορίζει την πηγή της εξωτερικής διακοπής σύμφωνα με τον
παρακάτω πίνακα. Η δεύτερη παράμετρος με τιμή pinChange μας ορίζει την ρουτίνα εξυπηρέτησης της
διακοπής ISR που δημιουργήσαμε παραπάνω. Η τρίτη παράμετρος με τιμές LOW, RISING, FALLING,
CHANGE μας ορίζει το είδος της αλλαγής λογικής στάθμης που ενεργοποιεί την διακοπή.

} // end of setup

void loop () // Η κύρια ρουτίνα του προγράμματος που εκτελείται συνεχώς
{
  // loop doing nothing
  // Δεν εκτελείται καμία εντολή αρά δεν γίνεται τίποτα και περιμένουμε να συμβεί μία διακοπή
}

```

## Below is a list of interrupts, in priority order, for the Atmega328:

1	Reset	
2	External Interrupt Request 0 (pin D2)	(INT0_vect)
3	External Interrupt Request 1 (pin D3)	(INT1_vect)
4	Pin Change Interrupt Request 0 (pins D8 to D13)	(PCINT0_vect)
5	Pin Change Interrupt Request 1 (pins A0 to A5)	(PCINT1_vect)
6	Pin Change Interrupt Request 2 (pins D0 to D7)	(PCINT2_vect)
7	Watchdog Time-out Interrupt	(WDT_vect)
8	Timer/Counter2 Compare Match A	(TIMER2_COMPA_vect)
9	Timer/Counter2 Compare Match B	(TIMER2_COMPB_vect)
10	Timer/Counter2 Overflow	(TIMER2_OVF_vect)
11	Timer/Counter1 Capture Event	(TIMER1_CAPT_vect)
12	Timer/Counter1 Compare Match A	(TIMER1_COMPA_vect)
13	Timer/Counter1 Compare Match B	(TIMER1_COMPB_vect)
14	Timer/Counter1 Overflow	(TIMER1_OVF_vect)
15	Timer/Counter0 Compare Match A	(TIMER0_COMPA_vect)
16	Timer/Counter0 Compare Match B	(TIMER0_COMPB_vect)
17	Timer/Counter0 Overflow	(TIMER0_OVF_vect)
18	SPI Serial Transfer Complete	(SPI_STC_vect)
19	USART Rx Complete	(USART_RX_vect)
20	USART, Data Register Empty	(USART_UDRE_vect)
21	USART, Tx Complete	(USART_TX_vect)
22	ADC Conversion Complete	(ADC_vect)
23	EEPROM Ready	(EE_READY_vect)
24	Analog Comparator	(ANALOG_COMP_vect)
25	2-wire Serial Interface (I2C)	(TWI_vect)
26	Store Program Memory Ready	(SPM_READY_vect)

## Δεν έχουν όλοι οι Arduino διακοπές στα ίδια pins

Most Arduino boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3). The table

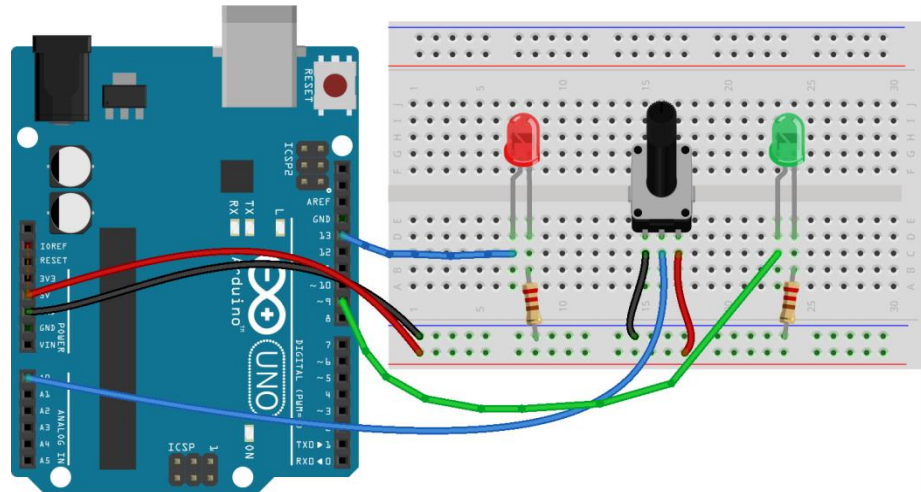
below shows the available interrupt pins on various boards.

Board	int.0	int.1	int.2	int.3	int.4	int.5
Uno, Ethernet	2	3				
Mega2560	2	3	21	20	19	18
Leonardo	3	2	0	1	7	
Due			(see below)			

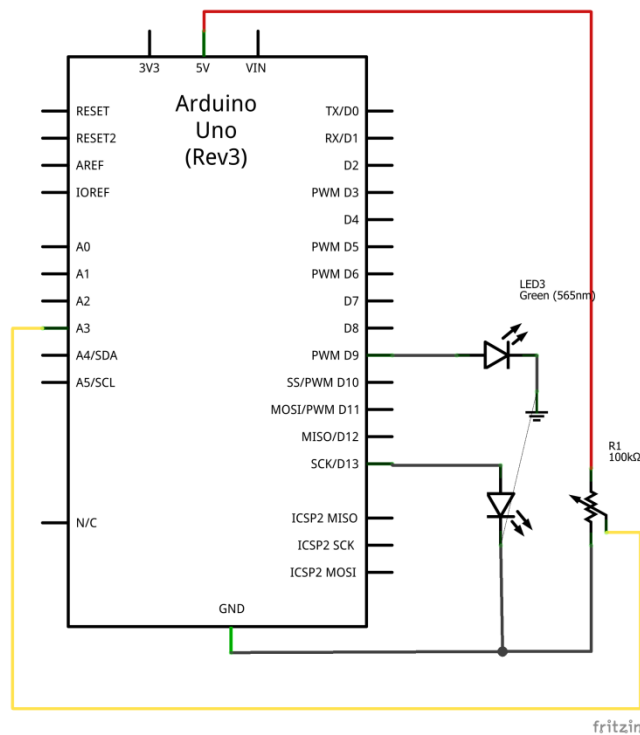
The Arduino Due board has powerful interrupt capabilities that allows you to attach an interrupt function on all available pins. You can directly specify the pin number in `attachInterrupt()`.

### 3.5 Εφαρμογή 4<sup>η</sup>

Διάβασμα αναλογικής τάσης, αναβόσβημα Led1 ανάλογα με την στάθμη της τάσης και έλεγχος φωτεινότητας Led2 μέσω PWM ανάλογα με την στάθμη της τάσης.



fritzing





Κώδικας:

```
int sensorPin = A0; // select the input pin for the potentiometer
```

// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα sensorPin και της δίνουμε την τιμή A0 (το A0 είναι μία ακέραια σταθερά και αντιστοιχεί σε έναν ακέραιο αριθμό που μας δείχνει το pin, του καναλιού 1 του ADC) με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε το ποτενσιόμετρο – αναλογική είσοδος.

```
int ledPin1 = 13; // select the pin for the LED1
```

// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα ledPin1 και της δίνουμε την τιμή 13 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε το led1.

```
int sensorValue = 0; // variable to store the value coming from the sensor
```

// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα sensorValue και της δίνουμε αρχικά την τιμή 0 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα για να αποθηκεύουμε τις τιμές που διαβάζει ο ADC

```
int ledPin2 = 9; // LED2 connected to digital pin 9
```

// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα ledPin2 και της δίνουμε την τιμή 9 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που θα συνδέσουμε το led2. – PWM Output

```
int outputValue = 0; // value output to the PWM (analog out)
```

// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα outputValue και της δίνουμε αρχικά την τιμή 0 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα για να δίνουμε τις αναλογικές τιμές φωτεινότητας για το led2

```
void setup() { // Η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
```

```
  // declare the ledPin1 as an OUTPUT:
```

```
  pinMode(ledPin1, OUTPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή ledPin1, ότι το pin 13 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V
```

```
  pinMode(ledPin2, OUTPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή ledPin2, ότι το pin 9 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V
```

```
}
```

```
void loop() { // Η κύρια ρουτίνα του προγράμματος που εκτελείτε συνεχώς
```

```
  // read the value from the sensor:
```

```
  sensorValue = analogRead(sensorPin); // Διάβασμα της αναλογικής τιμής της τάσης στο pin A0 του ADC και απόδοση της τιμής στην μεταβλητή sensorValue
```

```
  // turn the ledPin on
```

```
  digitalWrite(ledPin1, HIGH); // Βγάλε λογικό 1 (5V) στο pin 13, δηλαδή άναψε το led1
```

```
  // stop the program for <sensorValue> milliseconds:
```

```
  delay(sensorValue); // Συνάρτηση καθυστέρησης χρόνου σε ms ανάλογου με την τιμή της αναλογικής τάσης από 0 έως 1023 ms
```

```
  // turn the ledPin off:
```

```
  digitalWrite(ledPin1, LOW); // Βγάλε λογικό 0 (0V) στο pin 13, δηλαδή σβείσε το led1
```

```
  // stop the program for for <sensorValue> milliseconds:
```

```
delay(sensorValue); // Συνάρτηση καθυστέρησης χρόνου σε ms ανάλογου με
την τιμή της αναλογικής τάσης από 0 έως 1023 ms
```

```
//fade section
```

```
// map it to the range of the analog out:
```

```
outputValue = map(sensorValue, 0, 1023, 0, 255); // Συνάρτηση αντιστοίχισης –
μετατροπής των τιμών της αναλογικής τάσης από 0 – 1023 σε εύρος τιμών από 0-255
και απόδοση των νέων τιμών στην μεταβλητή outputValue
```

```
// change the analog out value:
```

```
analogWrite(ledPin2, outputValue); // Δημιουργία παλμών PWM στο pin 9 (led2)
ανάλογους με την τιμή του ADC με εύρος από 0 έως 255
```

```
// wait 2 milliseconds before the next loop
```

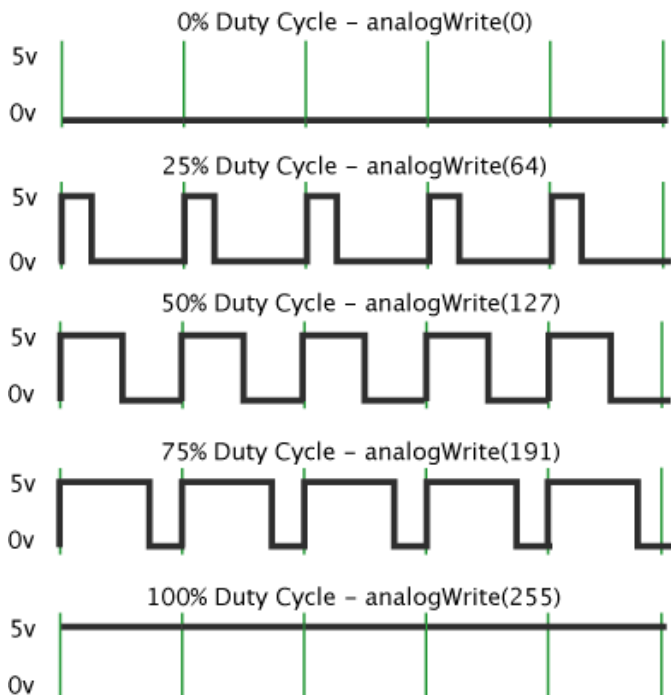
```
// for the analog-to-digital converter to settle
```

```
// after the last reading:
```

```
delay(2); // // Συνάρτηση καθυστέρησης χρόνου 2ms με σκοπό την επανεκκίνηση
του ADC
}
```

## Διαμόρφωση πλάτους παλμού

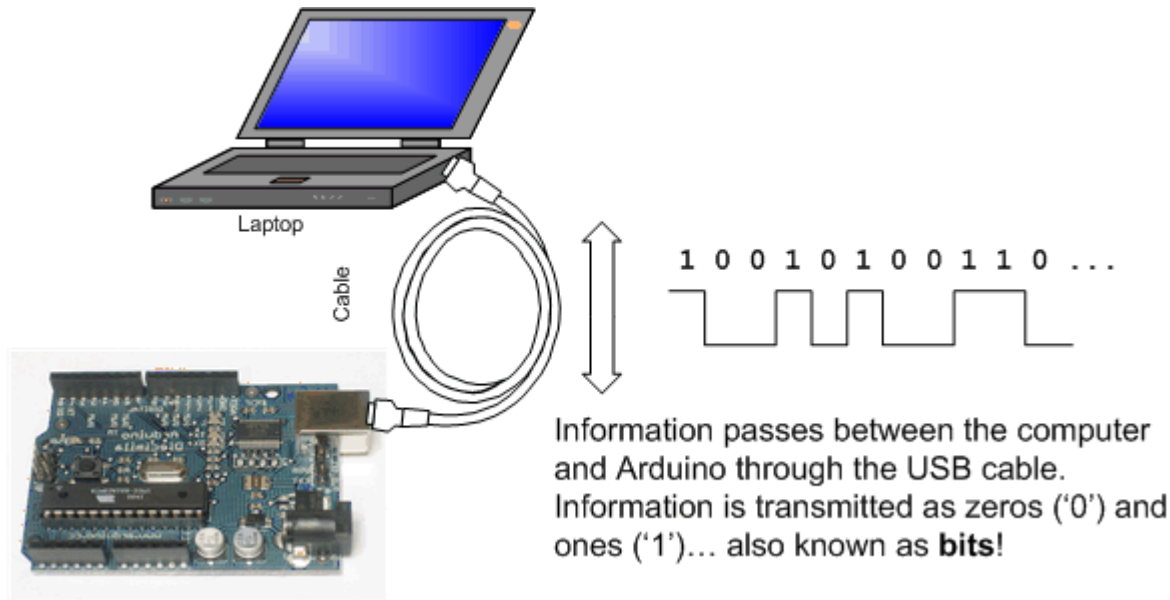
Στο παρακάτω γράφημα, οι πράσινες γραμμές αντιπροσωπεύουν μια κανονική χρονική περίοδο. Μια κλήση της `analogWrite()` είναι σε μια κλίμακα από 0 έως 255, έτσι ώστε για παράδειγμα, η κλήση `analogWrite(255)` ζητά το 100% του κύκλου λειτουργίας, και η `analogWrite(127)` το 50% του κύκλου λειτουργίας (στο μισό χρόνο) [12].



Διαμόρφωση πλάτους παλμού(PWM)

## 3.6 Εφαρμογή 5<sup>η</sup>

Διάβαση αναλογικής τάσης, ανάμνημα Led1 ή Led2 ανάλογα με την στάθμη της τάσης, και αποστολή δεδομένων σειριακά στον Η/Υ



Κώδικας:

```
int sensorPin = A0; // select the input pin for the potentiometer
// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα
sensorPin και της δίνουμε την τιμή A0 (το A0 είναι μία ακεραία σταθερά και
αντιστοιχεί σε έναν ακεραίο αριθμό που μας δείχνει το pin, του καναλιού 1 του ADC)
με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που
θα συνδέσουμε το ποτενσιόμετρο – αναλογική είσοδος.
int ledPin1 = 13; // select the pin for the LED1
// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα ledPin1 και της
δίνουμε την τιμή 13 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και
να δηλώσουμε που θα συνδέσουμε το led1.
int sensorValue = 0; // variable to store the value coming from the sensor
// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα sensorValue και της
δίνουμε αρχικά την τιμή 0 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον
κώδικα για να αποθηκεύουμε τις τιμές που διαβάζει ο ADC
int ledPin2 = 9; // LED2 connected to digital pin 9
// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα ledPin2 και της
δίνουμε την τιμή 9 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και
να δηλώσουμε που θα συνδέσουμε το led2.

int outputValue = 0; // value output
// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα outputValue και της
δίνουμε αρχικά την τιμή 0 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον
κώδικα
void setup() { // Η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή
// declare the ledPin1 as an OUTPUT:
pinMode(ledPin1, OUTPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή
ledPin1, ότι το pin 13 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V
```

```

pinMode(ledPin2, OUTPUT); // Ορίζουμε χρησιμοποιώντας την μεταβλητή
ledPin1, ότι το pin 9 θα λειτουργεί σαν ψηφιακή έξοδος 0-5V
// initialize serial communications at 9600 bps:
Serial.begin(9600); // Αρχικοποίηση της σειριακής επικοινωνίας με ρυθμό 9600
}

void loop() { // Η κύρια ρουτίνα του προγράμματος που εκτελείτε συνεχώς

// read the value from the sensor:
sensorValue = analogRead(sensorPin); // Διάβασμα της αναλογικής τιμής της
τάσης στο pin A0 του ADC και απόδοση της τιμής στην μεταβλητή sensorValue

// turn the ledPin on
if (sensorValue < 511 ) { // Εάν η τιμή του ADC είναι μικρότερη από 511 δηλαδή
2,5 volt τότε:
digitalWrite (ledPin1, HIGH); // Βγάλε λογικό 1 (5V) στο pin 13, δηλαδή άναψε το
led1
digitalWrite (ledPin2, LOW); // Βγάλε λογικό 0 (0V) στο pin 9, δηλαδή σβήσε το
led2

}
else //αλλιώς:
{
digitalWrite (ledPin1, LOW); // Βγάλε λογικό 0 (0V) στο pin 13, δηλαδή σβήσε το
led1

digitalWrite(ledPin2, HIGH); // Βγάλε λογικό 1 (5V) στο pin 9, δηλαδή άναψε το
led2

}

//serial section
// print the results to the serial monitor:
// Τμήμα σειριακής επικοινωνίας
Serial.print("\t sensor = " );
Serial.println(sensorValue);

// wait 2 milliseconds before the next loop
// for the analog-to-digital converter to settle
// after the last reading:
delay(2);
}

```

---

## 3.7 Εφαρμογή 6<sup>η</sup>

---

**Διάβασμα αναλογικής τάσης, μόνιμη αποθήκευση τιμών στην EEPROM, και αποστολή δεδομένων σειριακά στον Η/Υ διαβάζοντας την EEPROM**

Κώδικας:

```
#include <EEPROM.h> //Φόρτωση της βιβλιοθήκης με τις σχετικές συναρτήσεις για
επικοινωνία με την EEPROM

// the current address in the EEPROM (i.e. which byte
// we're going to write to next)
int addr = 0; // Ορίζουμε μία μεταβλητή τύπου ακεραίου με όνομα addr και αρχική τιμή
0 με σκοπό να την χρησιμοποιήσουμε για να καθορίζουμε την διεύθυνση εγγραφής στην
μνήμη
// start reading from the first byte (address 0) of the EEPROM
int address = 0; // Ορίζουμε μία μεταβλητή τύπου ακεραίου με όνομα address και αρχική
τιμή 0 με σκοπό να την χρησιμοποιήσουμε για να καθορίζουμε την διεύθυνση
ανάγνωσης από την μνήμη

byte value; // Ορίζουμε μία μεταβλητή τύπου byte με όνομα value για να αποθηκεύουμε
τα δεδομένα της μνήμης
int sensorPin = A0; // select the input pin for the potentiometer
// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα
sensorPin και της δίνουμε την τιμή A0 (το A0 είναι μία αέραια σταθερά και
αντιστοιχεί σε έναν αέραιο αριθμό που μας δείχνει το pin, του καναλιού 1 του ADC)
με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε που
θα συνδέσουμε το ποτενσιόμετρο – αναλογική είσοδος.

int sensorValue = 0; // variable to store the value coming from the sensor
// Ορίζουμε μία μεταβλητή τύπου Integer (Ακεραίου) με όνομα sensorValue και της
δίνουμε αρχικά την τιμή 0 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον
κώδικα για να αποθηκεύουμε τις τιμές που διαβάζει ο ADC

void setup() { // Η συνάρτηση αρχικών ρυθμίσεων του μικροελεγκτή

    Serial.begin(9600); // Αρχικοποίηση της σειριακής επικοινωνίας με ρυθμό 9600

}

void loop() { // Η κύρια ρουτίνα του προγράμματος που εκτελείτε συνεχώς

// read the value from the sensor:
sensorValue = analogRead(sensorPin);
); // Διάβασμα της αναλογικής τιμής της τάσης στο pin A0 του ADC και απόδοση
της τιμής στην μεταβλητή sensorValue
```

```

// need to divide by 4 because analog inputs range from
// 0 to 1023 and each byte of the EEPROM can only hold a
// value from 0 to 255.
int val = analogRead(sensorPin) / 4; //Ορισμός μίας τοπικής μεταβλητής τύπου
ακέραιου με όνομα val και απόδοση της αναλογικής τιμής της τάσης από το pin A0 του
ADC αφού διαιρεθεί με 4 αφού η μνήμη είναι 8 bit

// write the value to the appropriate byte of the EEPROM.
// these values will remain there when the board is
// turned off.
EEPROM.write(addr, val); // Εγγραφή της μνήμης στην διεύθυνση που μας δείχνει η
τιμή της μεταβλητής addr, την τιμή της μεταβλητής val

// advance to the next address. there are 512 bytes in
// the EEPROM, so go back to 0 when we hit 512.
addr = addr + 1; // Αύξηση της διεύθυνσης κατά ένα (επόμενη διεύθυνση)
if (addr == 512) // Εάν η διεύθυνση είναι η τελευταία αφού η EEPROM έχει
χωρητικότητα 512 byte, τότε:
    addr = 0; // Πήγαινε στην αρχική διεύθυνση

delay(100); // Καθυστέρηση 100ms

value = EEPROM.read(address); // Ανάγνωση της μνήμης στην διεύθυνση που μας
δείχνει η τιμή της μεταβλητής address, και απόδοση της τιμής στην μεταβλητή value

Serial.print(address); // Αποστολή σειριακά της τιμής της διεύθυνσης της μνήμης
Serial.print("\t");
Serial.print(value, DEC); // Αποστολή σειριακά της τιμής του περιεχομένου της
διεύθυνσης της μνήμης

Serial.println();

// advance to the next address of the EEPROM
address = address + 1; // Αύξηση της διεύθυνσης κατά ένα (επόμενη διεύθυνση)

// there are only 512 bytes of EEPROM, from 0 to 511, so if we're
// on address 512, wrap around to address 0
if (address == 512) // Εάν η διεύθυνση είναι η τελευταία αφού η EEPROM έχει
χωρητικότητα 512 byte, τότε:
    address = 0; // Πήγαινε στην αρχική διεύθυνση

delay(500); // Καθυστέρηση 500ms

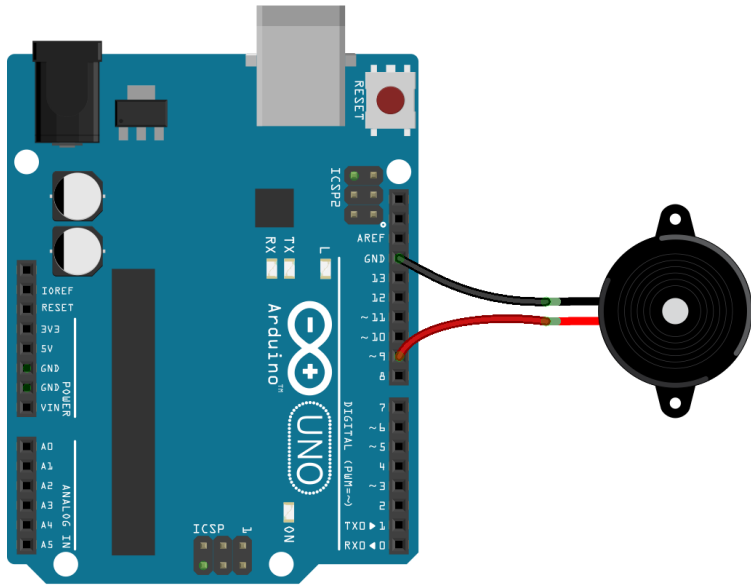
}

```

---

## 3.8 Εφαρμογή 7<sup>η</sup>

### Δημιουργία μουσικών τόνων «Twinkle Twinkle Little Star»



Piezoelectric speakers are frequently used as beepers in watches and other electronic devices. In this project, a piezo speaker is used to play melodies. It sends a square wave of the appropriate frequency to the piezo, generating the corresponding tone.

<http://arduino.cc/en/Tutorial/Melody>

fritzing

Κώδικας:

```
int speakerPin = 9; // Ορισμός μεταβλητής τύπου ακεραίου με όνομα speakerPin και
// απόδοση τιμής 9 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να
// δηλώσουμε που θα συνδέσουμε το buzzer
```

```
int length = 15; // the number of notes
// Ορισμός μεταβλητής τύπου ακεραίου με όνομα length και απόδοση τιμής 15 με
// σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και να δηλώσουμε τον
// αριθμό από νότες που θα έχει το μουσικό κομμάτι
char notes[] = "ccggaagffeeddc "; // a space represents a rest
// Ορισμός πίνακα με τιμές τύπου χαρακτήρα (char) και απόδοση τιμών
// ccggaagffeeddc τις νότες που θα έχει το μουσικό κομμάτι
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 }; // Ορισμός πίνακα με τιμές
// τύπου ακεραίου και απόδοση τιμών 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 για τις
// φορές που παίζετε η κάθε νότα που θα έχει το μουσικό κομμάτι
```

```
int tempo = 300; // Ορισμός μεταβλητής τύπου ακεραίου με όνομα tempo και
// απόδοση τιμής 300 με σκοπό να την χρησιμοποιήσουμε παρακάτω στον κώδικα και
// να δηλώσουμε τον ρυθμό (ταχύτητα) παιξίματος που θα έχει το μουσικό κομμάτι
```

```
void playTone(int tone, int duration) { // Ορισμός συνάρτησης με όνομα playTone
// και ακέραιες παραμέτρους tone (khz) και duration με λειτουργία την δημιουργία του
// τόνου μιας νότας
for (long i = 0; i < duration * 1000L; i += tone * 2) {
digitalWrite(speakerPin, HIGH);
delayMicroseconds(tone);
digitalWrite(speakerPin, LOW);
}
```

```

    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration) { // Ορισμός συνάρτησης με όνομα playNote
και παραμέτρους note και duration με λειτουργία το παίξιμο του τόνου μιας νότας

    char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' }; // Οι νότες
    int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };

    // play the tone corresponding to the note name
    for (int i = 0; i < 8; i++) {
        if (names[i] == note) {
            playTone(tones[i], duration); // Χρησιμοποιεί την προηγούμενη συνάρτηση
playTone
        }
    }
}

void setup() {
    pinMode(speakerPin, OUTPUT);
}

void loop() {
    for (int i = 0; i < length; i++) {
        if (notes[i] == ' ') {
            delay(beats[i] * tempo); // rest
        } else {
            playNote(notes[i], beats[i] * tempo);
        }

        // pause between notes
        delay(tempo / 2);
    }
}

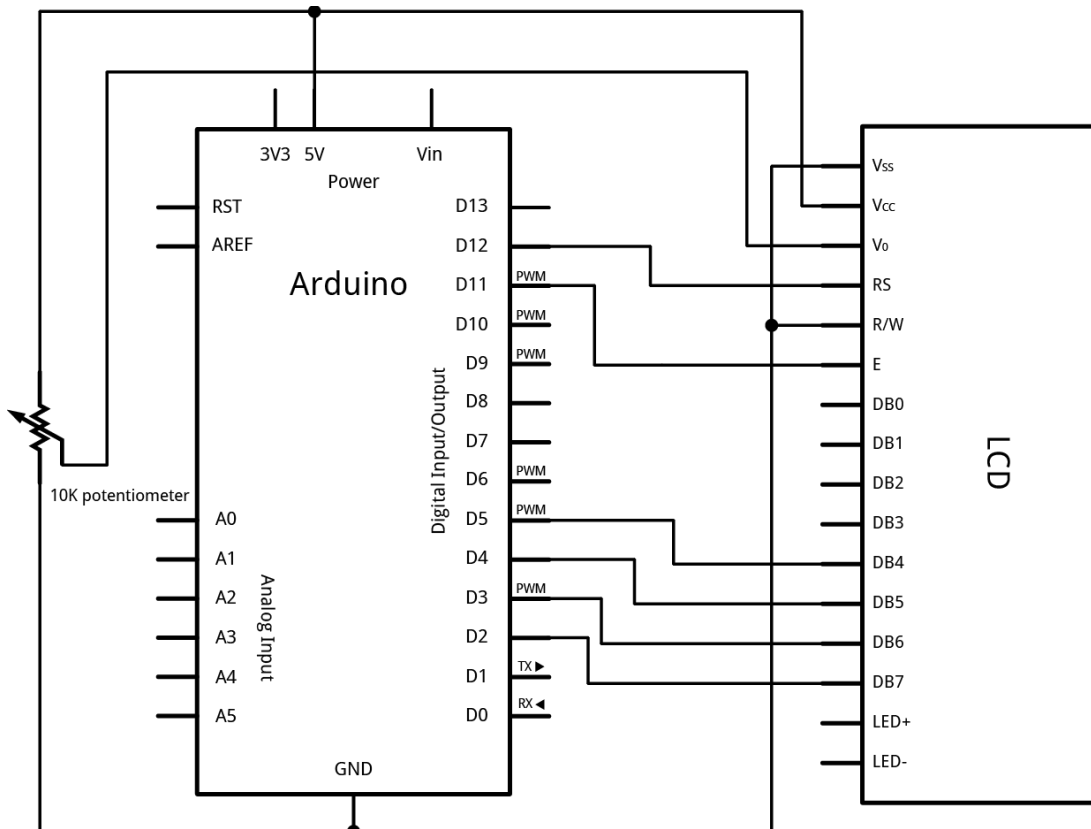
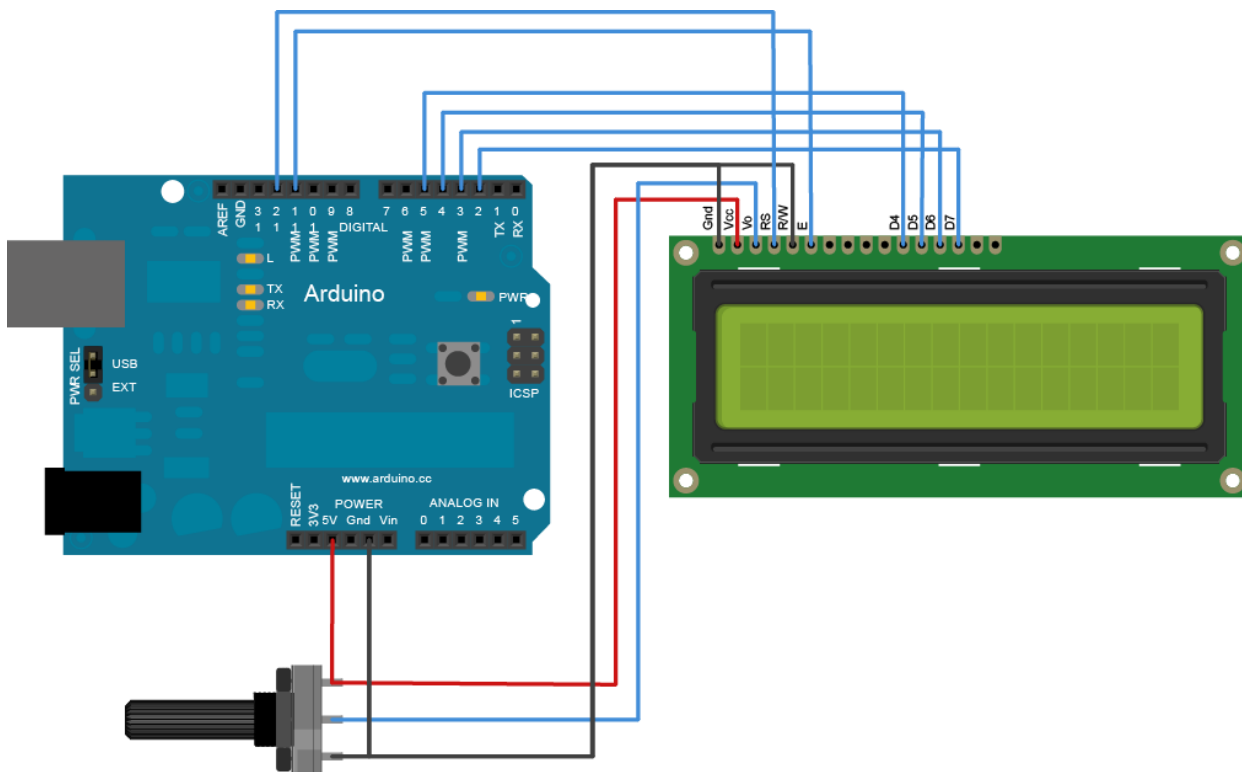
```

---



## 3.9 Εφαρμογή 8<sup>η</sup>

### Απεικόνιση χαρακτήρων σε Display LCD





Κώδικας:

```
/*
  LiquidCrystal Library - Hello World
```

Demonstrates the use a 16x2 LCD display. The LiquidCrystal library works with all LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.

This sketch prints "Hello World!" to the LCD and shows the time.

The circuit:

- \* LCD RS pin to digital pin 7
- \* LCD Enable pin to digital pin 6
- \* LCD D4 pin to digital pin 5
- \* LCD D5 pin to digital pin 4
- \* LCD D6 pin to digital pin 3
- \* LCD D7 pin to digital pin 2
- \* LCD R/W pin to ground
- \* 10K resistor:
  - \* ends to +5V and ground
  - \* wiper to LCD VO pin (pin 3)

Library originally added 18 Apr 2008  
 by David A. Mellis  
 library modified 5 Jul 2009  
 by Limor Fried (<http://www.ladyada.net>)  
 example added 9 Jul 2009  
 by Tom Igoe  
 modified 22 Nov 2010  
 by Tom Igoe

This example code is in the public domain.

```
http://www.arduino.cc/en/Tutorial/LiquidCrystal
*/
```

```
// include the library code:
#include <LiquidCrystal.h>
```

```
// initialize the library with the numbers of the interface pins
```

```
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
```

```
void setup() {
```

```
// set up the LCD's number of columns and rows:
```

```
lcd.begin(16, 2);
```

```
// Print a message to the LCD.
```

```
lcd.print("hello, world!");
```

```
}
```

```
void loop() {
```

```
// set the cursor to column 0, line 1
```

```
// (note: line 1 is the second row, since counting begins with 0):
```

```
lcd.setCursor(0, 1);
```

```
// print the number of seconds since reset:
```

```
lcd.print(millis() / 1000);
```

```
}
```

---

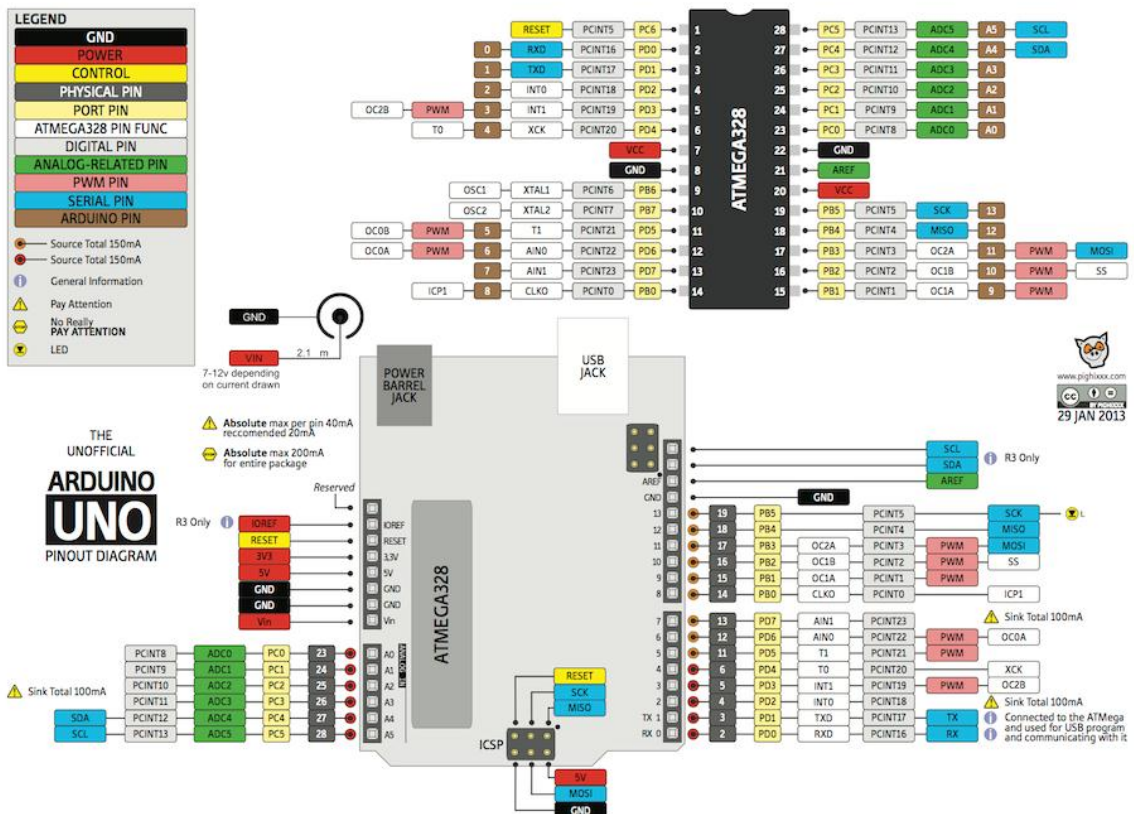
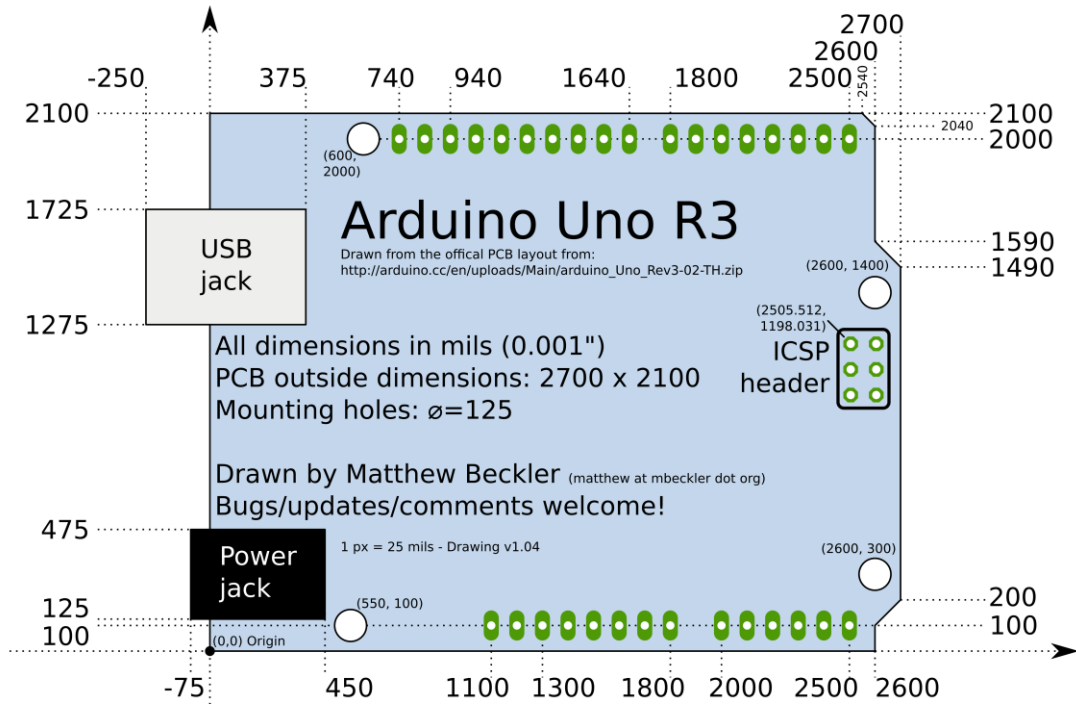
## Εφαρμογών συνέχεια....

---

- Ελεύθερη δημιουργία εφαρμογής από τους εκπαιδευόμενους.
- Επίδειξη σύνθετων εφαρμογών
- Τέλος Σεμιναρίου

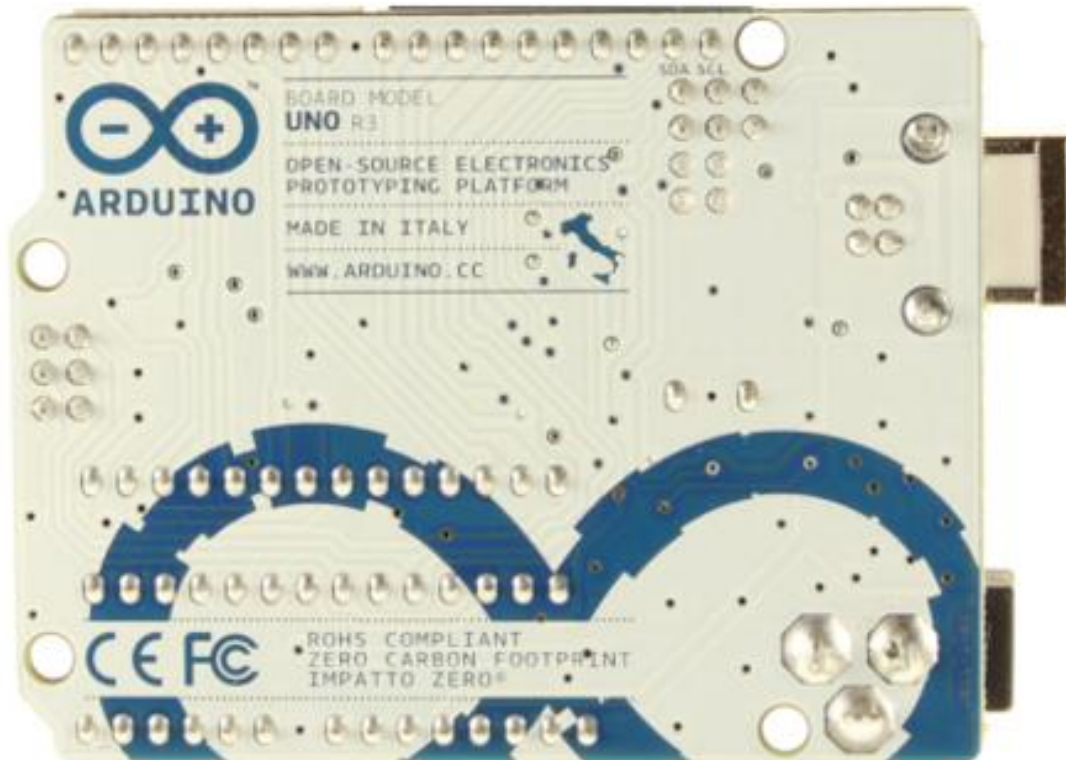


# Εικόνες - Σχέδια





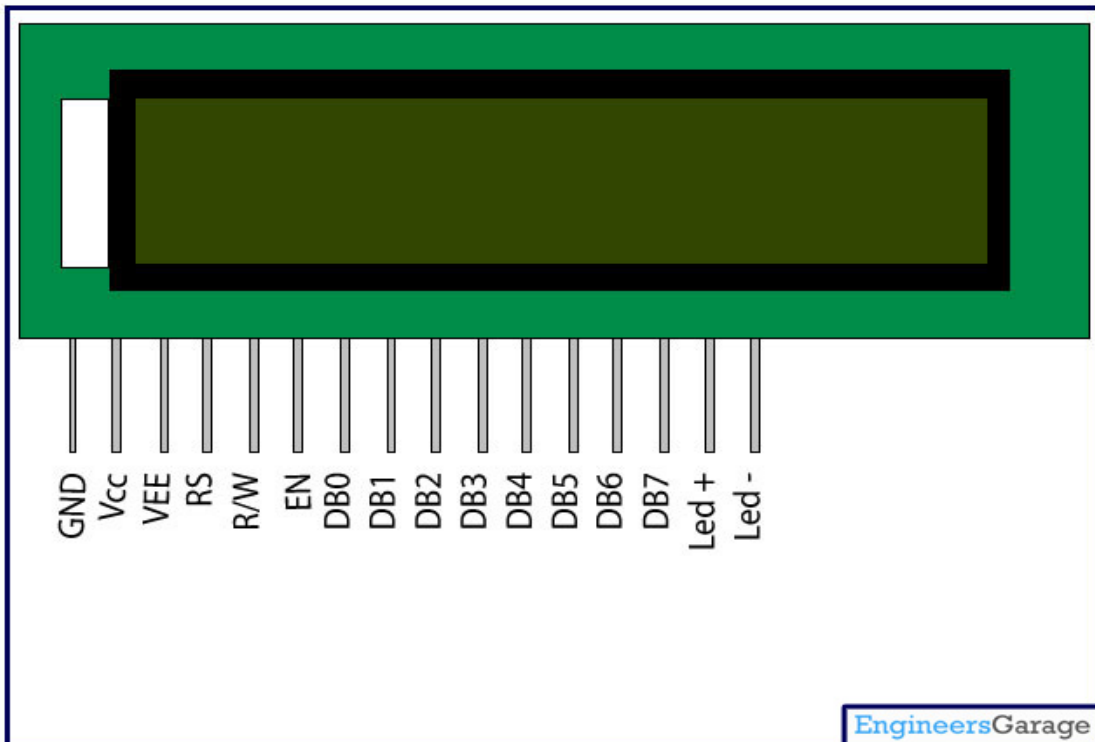
Εμπρός όψη πλακέτας arduino



Πίσω όψη πλακέτας arduino

## ATmega328

Arduino function					Arduino function
reset	(PCINT14/RESET) PC6	1	28	PC5 (ADC5/SCL/PCINT13)	analog input 5
digital pin 0 (RX)	(PCINT16/RXD) PD0	2	27	PC4 (ADC4/SDA/PCINT12)	analog input 4
digital pin 1 (TX)	(PCINT17/TXD) PD1	3	26	PC3 (ADC3/PCINT11)	analog input 3
digital pin 2	(PCINT18/INT0) PD2	4	25	PC2 (ADC2/PCINT10)	analog input 2
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5	24	PC1 (ADC1/PCINT9)	analog input 1
digital pin 4	(PCINT20/XCK/T0) PD4	6	23	PC0 (ADC0/PCINT8)	analog input 0
VCC	VCC	7	22	GND	GND
GND	GND	8	21	AREF	analog reference
crystal	(PCINT6/XTAL1/TOSC1) PB6	9	20	AVCC	VCC
crystal	(PCINT7/XTAL2/TOSC2) PB7	10	19	PB5 (SCK/PCINT5)	digital pin 13
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11	18	PB4 (MISO/PCINT4)	digital pin 12
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12	17	PB3 (MOSI/OC2A/PCINT3)	digital pin 11(PWM)
digital pin 7	(PCINT23/AIN1) PD7	13	16	PB2 (SS/OC1B/PCINT2)	digital pin 10 (PWM)
digital pin 8	(PCINT0/CLKO/ICP1) PB0	14	15	PB1 (OC1A/PCINT1)	digital pin 9 (PWM)



MECHANICAL DATA						PIN CONNECTIONS			
Item	Nominal Dimensions			Unit	PIN	Symbol	Level	Function	
Module Size(W×H×T)	80.0×36.0×12.0			mm	1	VSS	—	GND(0V)	
Viewing Area(W×H)	64.0×14.0			mm	2	VDD	—	Supply Voltage for Logic(+5V)	
Character Size(W×H)	3.00×5.02			mm	3	V0	—	Power supply for LCD	
Dot Size(W×H)	0.52×0.54			mm	4	RS	H/L	H: Data; L: Instruction Code	
					5	R/W	H/L	H: Read; L: Write	
					6	E	H/L	Enable Signal	
					7	DB0	H/L	Data Bus Line	
					8	DB1	H/L		
					9	DB2	H/L		
					10	DB3	H/L		
					11	DB4	H/L		
					12	DB5	H/L		
					13	DB6	H/L		
					14	DB7	H/L		
					15	LEDA	H/L	Backlight Power(+5V)	
					16	LEDK	—	Backlight Power(0V)	

ABSOLUTE MAXIMUM RATINGS					
Item	Symbol	Min	Type	Max	Unit
Operating Voltage	VDD	4.5	5.0	5.5	V
Operating Current	IDD	1.1	1.3	1.6	mA
LED Voltage	V <sub>LED</sub>	4.5	5.0	5.5	V
LED Current	I <sub>LED</sub>	19	20	21	mA
Operating Temp.	Topr	-20	—	+70	°C
Storage Temp.	Tsto	-30	—	+80	°C

ELECTRICAL CHARACTERISTICS					
Item	Symbol	Min	Type	Max	Unit
Input High Voltage	V <sub>IH</sub>	2.2	—	VDD	V
Input Low Voltage	V <sub>IL</sub>	0	—	0.6	V
Output High Voltage	V <sub>OH</sub>	2.4	—	VDD	V
Output Low Voltage	V <sub>OL</sub>	0	—	0.4	V

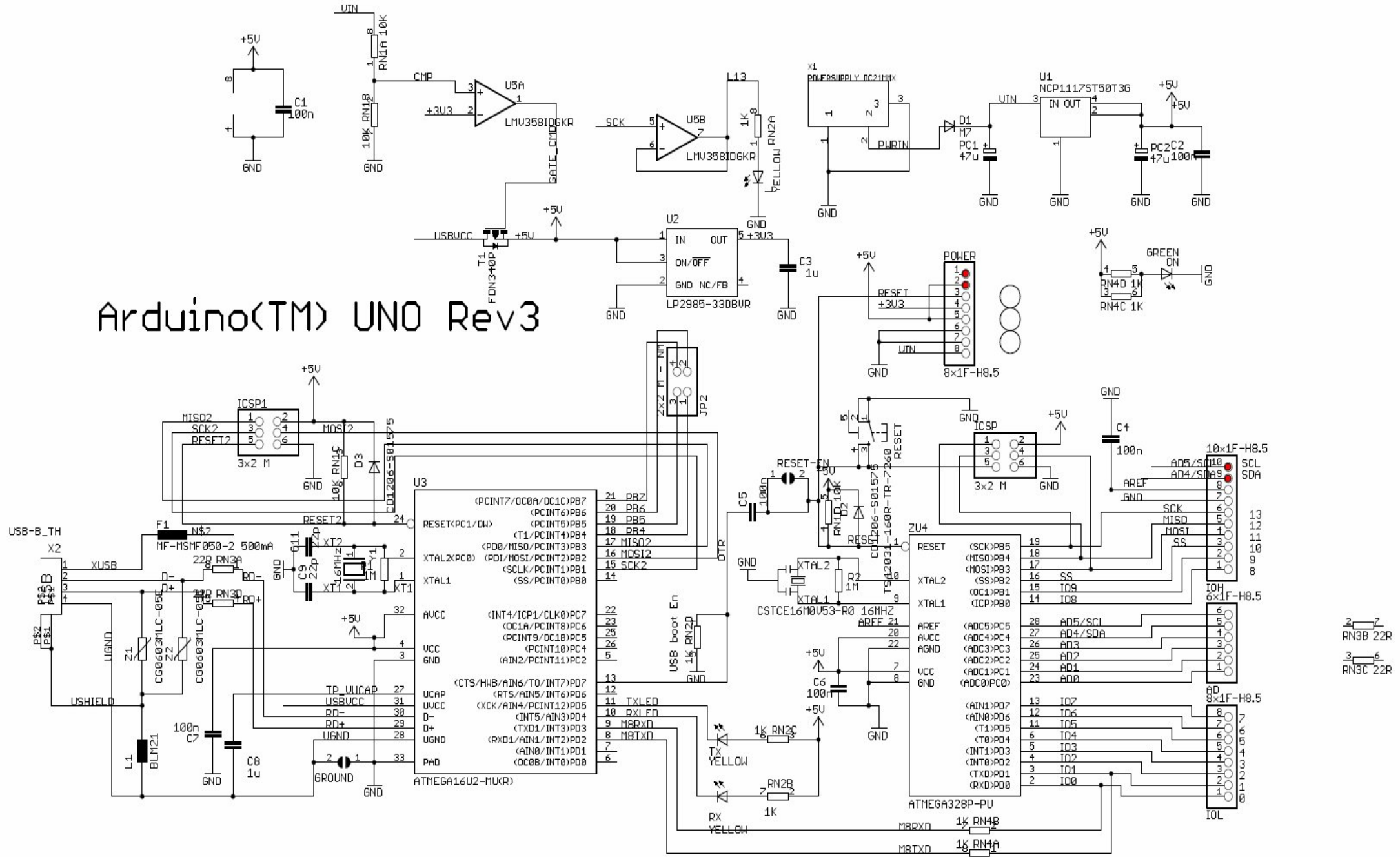
BLOCK DIAGRAM	

POWER SUPPLY	

Hitachi HD44780 driver

# Arduino(TM) UNO Rev3



2 7  
RN3B 22R  
3 6  
RN3C 22R